[redacted]

510.84
Il6N
no.623
cop.2

Math

UIUCDCS-R-74-623

OPTIMAL NETWORKS WITH NOR-OR GATES AND WIRED-OR LOGIC

by

Tsuneo Kawasaki

January 1974

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-74-623

OPTIMAL NETWORKS WITH NOR-OR GATES AND WIRED-OR LOGIC

by

Tsuneo Kawasaki

January 1974

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. GJ-40221 and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, May 1974.

ACKNOWLEDGEMENT

The author would like to express his deep gratitude to his advisor, Professor Saburo Muroga, for his enthusiastic guidance and encouragement during the preparation of this thesis, and also for his careful reading and valuable suggestions for the improvement of the original manuscript.

The author wishes to thank Dr. Y. Kambayashi for his many suggestions. He also thanks his colleague, J. N. Culliney, H. C. Lai and K. R. Hohulin for their variable suggestions.

This work was supported in part by the National Science Foundation under Grant No. NSF-GJ-40221.

ABSTRACT

Generally, Emitter Coupled Logic (ECL) gates have dual outputs (i.e., NOR-OR) and the outputs of two or more ECL gates can be tied together to realize OR functions without extra gates. This is called Wired-OR. Using gates with dual outputs and Wired-ORs, an algorithm to get the optimal networks, i.e., those which have a minimum number of NOR-OR gates and, as the secondary objective, a minimum number of connections, for a given arbitrary function, is discussed in this paper, under the assumption that only non-complimented variables are available as the network inputs. Only NOR-OR gates are used in these networks, but this algorithm can also be applied to networks with NAND-AND gates and Wired-ANDs.

First, the algorithm derives all networks which have a minimum number of NOR gates, using NOR gates only, and next it obtains all possible OR outputs of gates in the NOR network such that OR output can be connected to a successor gate without changing its output valve but with replacing its inputs by this connection. Finally, it forms Wired-ORs in the resultant network with NOR-OR gates. Thus the optimal network for the given function has been obtained.

Based on this algorithm, optimal networks for all functions of three variables and also some functions of four variables are found.

TABLE OF CONTENTS

1. INTRODUCTION

One of the major objectives in logical design of digital networks is the derivation of optimal networks. The optimality of a network is usually defined as the minimization of the number of gates, connections, levels, the chip area for an integrated circuit (IC), or others.

In this paper, using Emitter Coupled Logic (ECL) gates in ICs which have NOR-OR functions as their dual outputs and assuming that all external variables are non-complemented, logical design problem of optimal network is discussed, permitting the use of Wired-OR logic defining the optimality as the minimization of the number of gates first, and the number of connections next. Also optimal networks with ECL gates are derived for all switching functions of three variables and also functions of four variables which can be implemented with five or less NOR gates. The algorithm for this logical design problem consists of three phases as follows:

Phase 1: All the networks consisting of only NOR gates in which the number of NOR gates is minimized without necessarily minimizing the number of connections are derived by Y. Kambayashi using the logical design procedure based on Integer Programming (IP) formulation developed by Muroga et. al.(2)~(5),(8)~(12) In this phase, all the NOR networks for each function are synthesized. The basic nature of the IP formulation is to model the network using linear inequalities where the unknowns in these inequalities assume the value 0 or 1, which represent the absence or presence of a connection, respectively. Each solution of these inequalities corresponds to a network. This is a 0-1 IP problem.

Phase 2: All the OR output of gates such that the OR output of each gate can be connected to a successor gate without changing its output value but with replacing its inputs corresponding to that OR output are exhausted by checking every pair of the OR output of a gate and a successor gate in each NOR network obtained in phase 1. Then the connections from those OR outputs actually replace the corresponding input connections in the successor gate in the original network in order to reduce the number of connections in the network. (See Figure 2.1 where the connections to gate k from the inputs to gate j in (a) are replaced by the OR output of gates j, as shown in (b)).

Phase 3: Wired-ORs which are formed by tying down some connections without changing the output values of gates to which these connections go are exhausted by checking every set of connections in each network obtained in Phase 2. Then the optimal networks for each function has been exhausted.

Wired-ORs are formed as follows: If output of some gates and/or some external variables are connected to only the same set of gates, (Figure 1.1 (a)), we can form a Wired-OR as shown in Figure 1.1 (b).

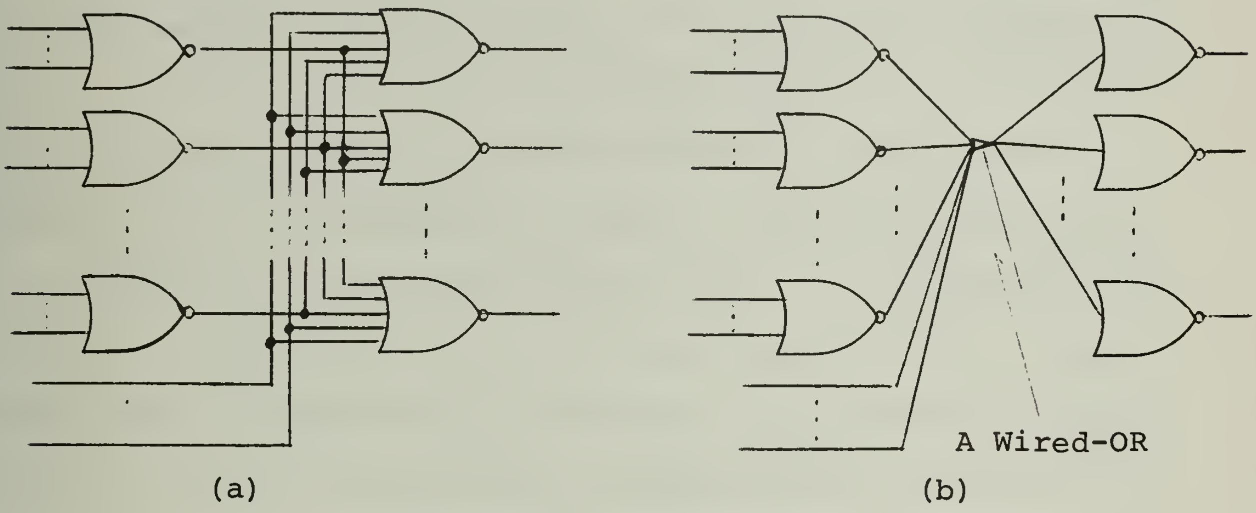


Figure 1.1--A Wired-OR

Compared with the design approach which derives the optimal network of NOR-OR gates with Wired-ORs by the IP formulation only, this approach reduces greatly computer time. It took 200 seconds (IBM 360/75) for all switching functions (77 representative functions by P-equivalence excluding trivial functions) of three variables, and 130 seconds for functions (312 representative function by P-equivalence) of four variables which can be implemented with five or less NOR-OR gates.

2. BASIC PROPERTIES AND PROCEDURE OUTLINE

Let us describe basic properties in each phase in more details.

2.1 Phase 1: Solution of the logical design problems based on the IP formulation.

In Phase 1, all networks which have the minimum number of NOR gates but where a number of connections is not necessarily minimized (i.e., all networks with a minimum number of gates no matter whether they have a minimum number of connections as the secondary objective) are obtained for each function by solving the logical design problem based on the IP formulation which is discussed in the next chapter.

2.2 Phase 2: Derivation of OR-output-to-gate connections.

(A) In each of the networks derived in Phase 1, gates generally have inputs from NOR outputs of the other gates and/or external variables. If gates j and k share the same inputs like Figure 2.1(a), the three inputs to gate k may be

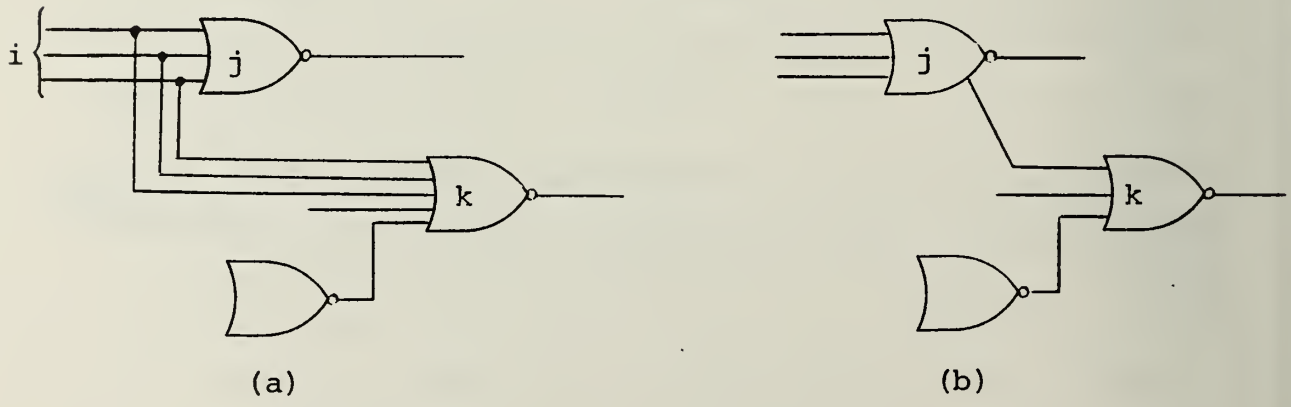


Figure 2.1--OR-output-to-gate connection

replaced by a single connection from the OR output of gate j like Figure 2.1(b). Generally i-1 connections are eliminated if two gates j and k share i inputs in Figure 2.1. If gate j has only one input (i.e., i = 1), the number of connections does not change. But this connection is still a candidate of OR-output-to-gate connection by the following reason. When we consider Wired-ORs, the OR output corresponding to this input can be an input to a Wired-OR which is to be considered in Phase 3. (See Figure 2.2).

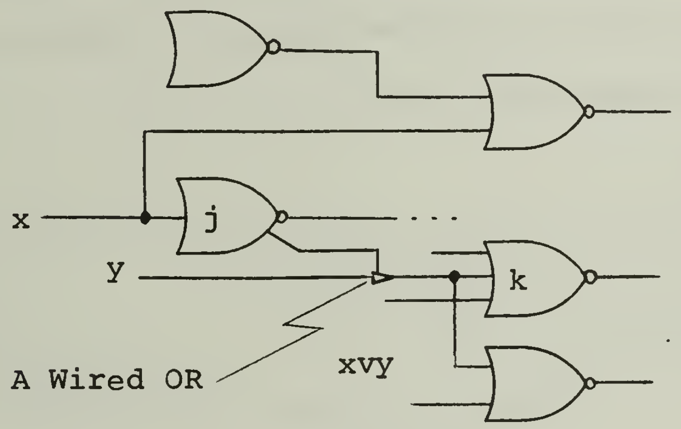


Figure 2.2--The OR output from a single input gate

(B) A gate k can have inputs from the OR outputs of other gates under the following three restrictions.

<u>Restriction 1</u>:

(i) Gate k must have two or more inputs.

If gate k has only one input, in other words, the OR output of another gate j is connected to gate k like Figure 2.1(b) to replace the connection from the input of gate j to gate k, these two gates have the same output functions, (i.e., NOR and OR outputs) and one gate is redundant. (See Figure 2.3).

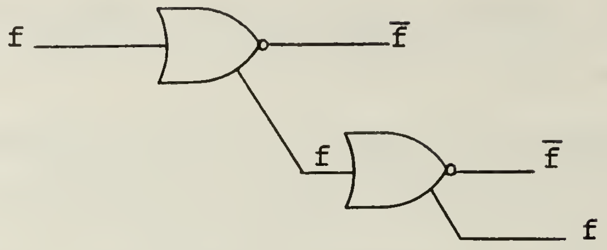


Figure 2.3--A redundant gate

(ii) Inputs of gate k must not be connected from the OR outputs of its successor gates or itself to avoid a loop (See Figure 2.4).

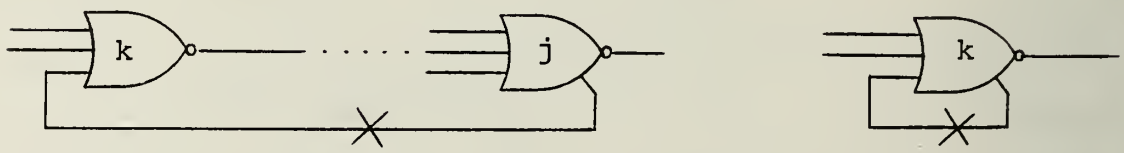


Figure 2.4--Loop-free networks

(iii) Inputs of gate k must not be connected from other gates whose NOR outputs are already connected to gate k. This is because the output of gate k becomes always "O" (Figure 2.5).

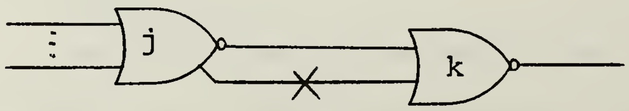


Figure 2.5--The Output of gate k is always "O"

(C) All the possible OR-output-to-gate connections are derived by checking every pair of the OR output of a gate and a successor gate such as gates j and k in Figure 2.1(b).

Some gates have a possibility to have only one input from an OR output, and some other gates have possibilities to have two or more inputs from OR outputs. But in all of these cases we do not necessarily replace the input connections to the successor gate by the corresponding OR-output-to-gate connections to reduce the number of connections. The reason for this is as follows: If a gate k_s has a possibility to have only one input from the OR output of gate j_s, and if this OR output is inputs to other gates as shown in Figure 2.6(a), we

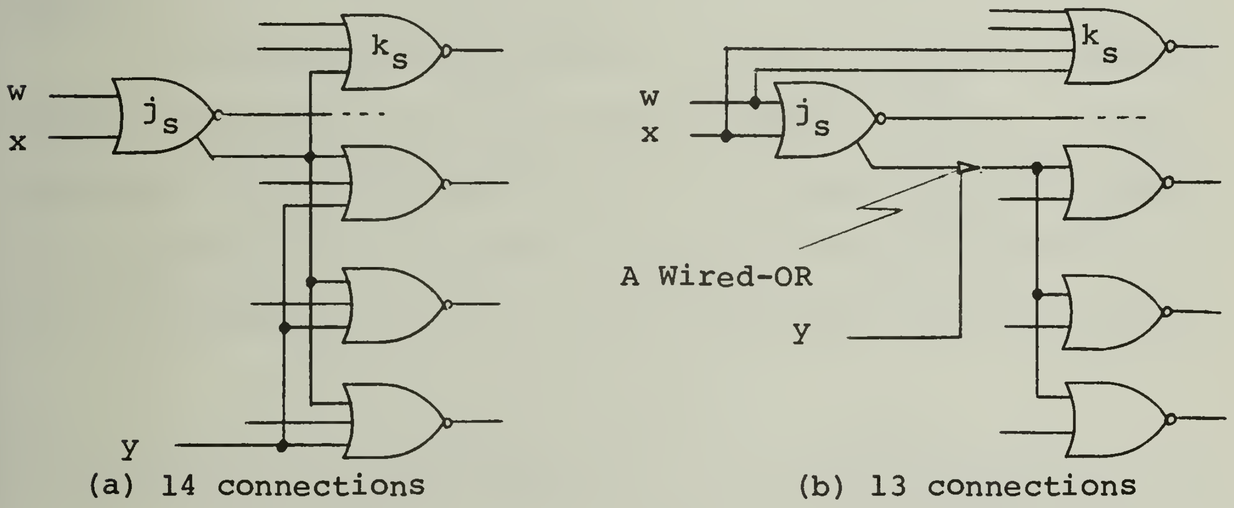
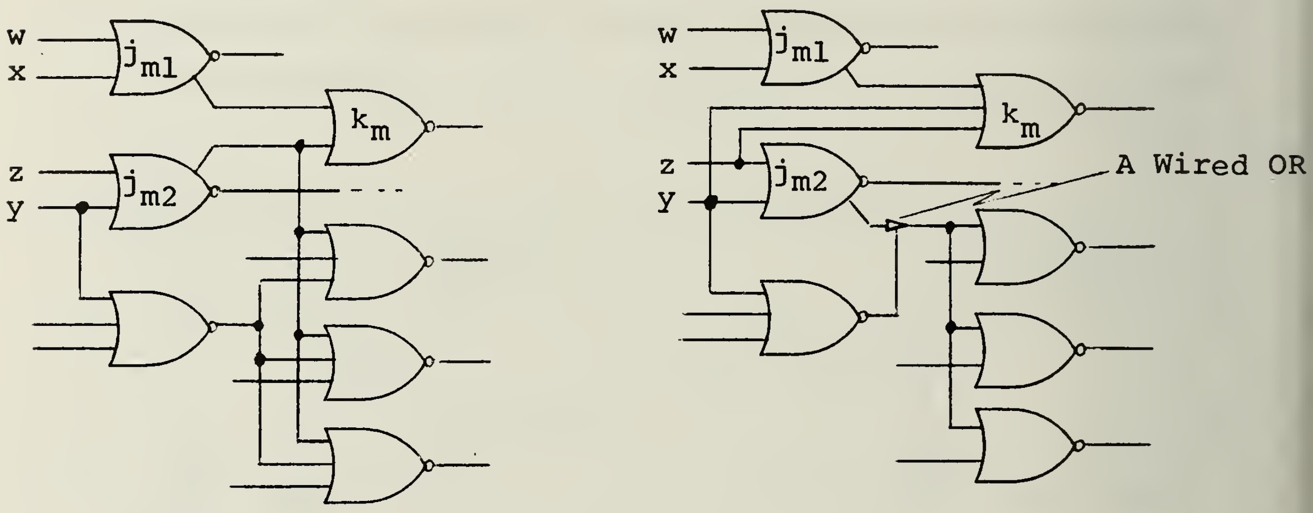


Figure 2.6--Some gate should keep original inputs

can derive a Wired-OR for the inputs to the other three successor gates unless we replace the inputs to gate k_s by OR-output-to-gate connection from gate j_s as shown in Figure 2.6(b). The number of connections in the network in Figure 2.6(b) is smaller than that in (a). So gate k_s should not have an input from OR output but have original inputs. Consequently, we have to further check every OR-output-to-gate connection later in section 2.3 (B) to find out whether a Wired-OR instead of the OR-output-to-gate connections can reduce the number of connections.

If a gate has possibilities to have two or more inputs from OR outputs, these OR-output-to-gate connections have the same problem, as illustrated in Figure 2.7. If gate k_m has two



(a) Connections in class i
18 connections

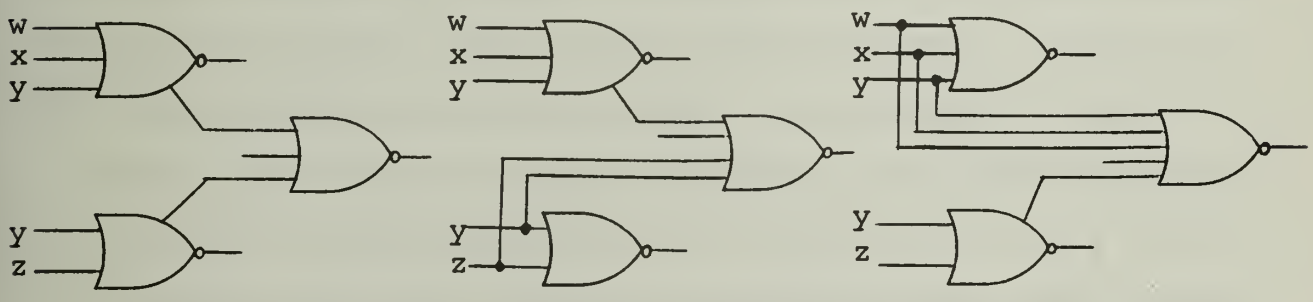
(b) Connections in class i_s
17 connections

Figure 2.7--The set in class is should be chosen

inputs from OR outputs of gates j_{m1} and j_{m2}, a Wired-OR cannot be formed for inputs of the other three gates (See Figure 2.7(a)). But if gate k_m has only one input from the OR output of gate j_{m1}, a Wired-OR can be formed for inputs of the other three gates as shown in Figure 2.7(b). This reduces the number of connections in the entire network, though the number of inputs of gate k_m increases.

Therefore we divide all the gates in the network into three sets of gates S, M, and the remainder in order to facilitate our checking OR-output-to-gate connections to find out whether Wired-ORs can reduce the number of connections. Set S consists of gates each of which has a possibility to have only one input from an OR output like gate k_s in Figure 2.6 (a), and set M consists of gates each of which has possibilities to have two or more inputs from OR outputs like gate k_m in Figure 2.7.

If a gate has a possibility to have Ro inputs from OR outputs, then there are $2^{Ro}-1$ possible ways to replace the corresponding inputs of the gate by the OR-output-to-gate connections, where Ro is the number of the OR outputs. Examples are shown in Figure 2.8 for Ro =2, and there are $2^2-1 = 3$ possible



(a) class 1 (b) class 2 (c) class 3

Figure 2.8--All possible ways to replace by two OR-output-to-gate connections ways to replace the corresponding inputs by OR-output-to-gate connections like (a), (b), and (c). If we do not consider Wired-ORs, the connection configuration in (a) should be chosen because the number of connections is the minimum among these three connection configurations, (a), (b) and (c). But when we consider Wired-ORs as we will in the next section, some OR-output-to-gate connections must not replace the corresponding inputs like the case of the OR output of gate j_{m2} in Figure 2.7 (b), even though the number of inputs of gate k_m increases. Since the above trade off between Wired-ORs and OR-output-to-gate connections differs with different connection configurations, we classify these $2^{Ro}-1$ connection configurations according to the number of all the input connections to gate k_m, as follows; class 1 is the configurations in each of which gate k_m has the minimum number of connections (i.e., the minimum compared with the numbers of connections of other connection configurations of the gate k_m), class 2 is the configuration in each of which gate k_m has the second minimum of connections, and so on. Thus in Figure 2.7 a set of connections in class i_s should be chosen rather than a set of connections in class i, though $i < i_s$. Thus we have to consider which class of connections has the minimum number of connections in the entire network, by taking into account Wired-ORs. $2^{Ro}-1$ is a fairly large number even if Ro is small, and it is tedious to derive Wired-ORs.

Suppose gate k_m has different configurations of connections from OR outputs and these configurations are different classes. Then if we keep only connection configurations satisfying the next restriction discarding other connection configurations, we can decrease the number of connection configurations to be considered such that the scope of later search for optimal networks is narrowed down.

Restriction 2:

(i) Choose a connection configuration in class 1 if every connection configuration has no OR outputs connected to any Wired-OR. This is because we do not need to take into account Wired-ORs in this case, and class 1 is a configuration in which gate k_m has the minimum number of connections.

(ii) Otherwise, choose every connection configuration which has OR outputs connected to any Wired-OR and also satisfy $d > 0$ discussed in the following. This is because if a connection configuration in class i $(i > 1)$ has OR outputs connected to any Wired-OR, there is a possibility that the number of connections in the connection configuration in class i becomes smaller with Wired-ORs than the number of connections in the connection configuration in class 1.

The number of connections for a Wired-OR with k_1 inputs and k_2 outputs is counted as $k_1 + k_2 - 1$. (We will discuss in the next section why the number of connections for a Wired-OR is counted as $k_1 + k_2 - 1$.) If we do not use this Wired-OR, $k_1 \times k_2$ connections are required. So we can reduce the number of connections by at most $k_1 \times k_2 - (k_1 + k_2 - 1)$ using this Wired-OR. But the number of connections to gate k_m increases by using this connection configuration in class i instead of the connection configuration in class 1. Let us assume this increase to be k_c. Let us define d as:

$$d = k_1 \times k_2 - (k_1 + k_2 - 1) - k_c$$

d is the maximum number by which we can reduce the number of connections in the network by using this connection configuration in class i instead of the connection configuration in class 1. Thus, if the number of connections is reduced by Wired-ORs, $d > 0$ must hold.

2.3 Phase 3: Derivation of Wired-ORs

Let us discuss basic properties in (A) and (B) first.

(A) The following properties about the effects of use of Wired-ORs are known.[6] The networks obtained by Phase 1 consist of only NOR gates and Phase 2 was applied to all NOR gates except the network output gate (gate 1). First notice that Phase 2 obviously does not change the number of gates, though the number of connections may be reduced. Theorem 1 shows that the number of gates is still not reduced even when Wired-ORs are considered to all NOR-OR gates except the network output NOR gate.

Theorem 2 shows a basic property when a Wired-OR is considered to only the network output NOR gate, if the network output gate has the NOR output only.

Theorem 1 The number of NOR-OR gates in any network consisting of NOR-OR gates only without Wired-ORs which has a minimum number of NOR-OR gates can not be reduced by using Wired-ORs whose outputs are connected to NOR-OR gates, but not using Wired-ORs whose outputs are connected to the network output terminal. The number of connections in the network may be reduced.

Proof Assume that there exists a network which has fewer NOR-OR gates by mixing with Wired-ORs whose outputs are connected to NOR-OR gates than a network without Wired-ORs which has a minimum number of gates. Change all the Wired-ORs such as (a) in Figure 2.9 to ordinary input connections in (b). In this conversion, no NOR-OR gate is added but some connections are added, so the new network without

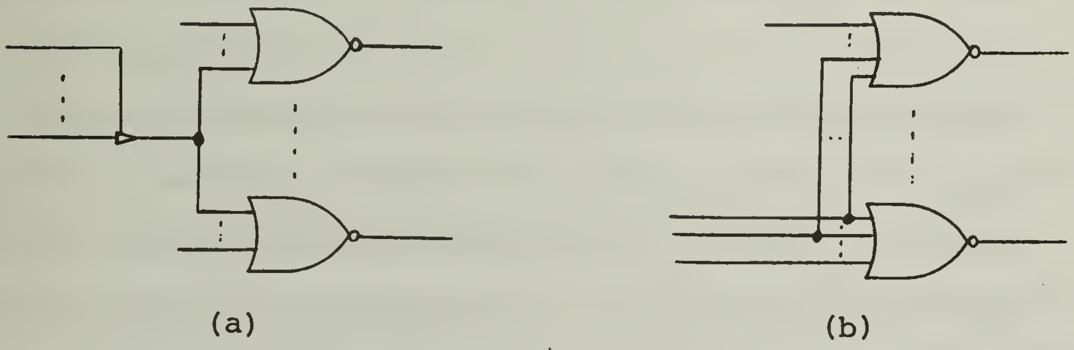


Figure 2.9--Conversion of a NOR-OR network

Wired-ORs have fewer NOR-OR gates than the network with a minimum number of NOR-OR gates without Wired-ORs. This contradicts the assumption that the network with a minimum number of NOR-OR gates without Wired-ORs has the minimal number of gates. The number of connections may be reduced by going from (b) to (a). Q.E.D.

Because of Theorem 1, the number of gates is not changed by Part 1 of Phase 3. But since the reduction of the number of connections is different for a different function in Phase 2 or 3, we considered all networks with a minimum number of NOR gates but without minimizing the number of connections in Phase 1.

Theorem 2 If an optimal NOR-OR network without Wired-ORs realizing function f consists of $R_n - 1$ NOR-OR gates and if the network output gate has the NOR output only, an optimal network realizing the same function using NOR-OR gates and only a Wired-OR connected to the network output No Wired-ORs connected to any other gate) may have at most two less gates, the number of connections may also be reduced, and there exist functions which consist of exactly $R_n - 2$ NOR-OR gates in their optimal networks.

Proof If optimal networks without Wired-ORs for f and $\overline{f}$ require R_{n1} and R_{n2} NOR-OR gates (including the network output gate which has the NOR output only), respectively then $|R_{n1} - R_{n2}| \leq |$ holds, because a network realizing

f or $\bar{f}$ can be obtained by connecting one extra NOR gate to the network output gate of an optimal network without Wired-ORs realizing $\bar{f}$ or f, respectively. Suppose the network for f without Wired-ORs has $R_{n1} = R_{n2} + 1$ NOR-OR gates (including the network output gate which has the NOR output only). The network realizing f is obtained by replacing the network output gate of an optimal network for $\bar{f}$ taking into account a Wired-OR, and this network requires $R_{n2} - 1$ NOR-OR gates. This means that the network for f requires $R_{n1} - 2$ NOR-OR gates. Since Wired-ORs other than the Wired-OR whose output is connected to the network output do not reduce the number of NOR-OR gates as stated in Theorem 1, this number $R_{n1} - 2$ is the least possible result. Q.E.D.

Theorem 1 is still true even if "NOR-OR" in the statement is replaced by "NOR."

Since Wired-ORs are not ordinary gates physically, Wired-ORs shown in Figure 1.1 should satisfy the following restrictions.(6)

Restriction 3:

(i) If an input of a NOR-OR gate is connected to a Wired-OR, that input cannot be connected to other gate, other Wired-ORs or the output terminal.

(ii) If an external variable is connected to a Wired-OR, it cannot be connected to any gates, any other Wired-ORs or the output terminal.

(iii) A Wired-OR can not be connected to any other Wired-ORs.

(B) Next, we define how to count the number of connections for a Wired-OR.

Suppose we have a network shown in Figure 2.10 (a). Out of this connection configuration, we can form Wired-ORs as shown in Figure 2.10 (b) and (c). Each Wired-OR in these examples has three inputs and goes to three gates. From the original network (a) which requires nine connections, we get network (b) consisting of six connections.

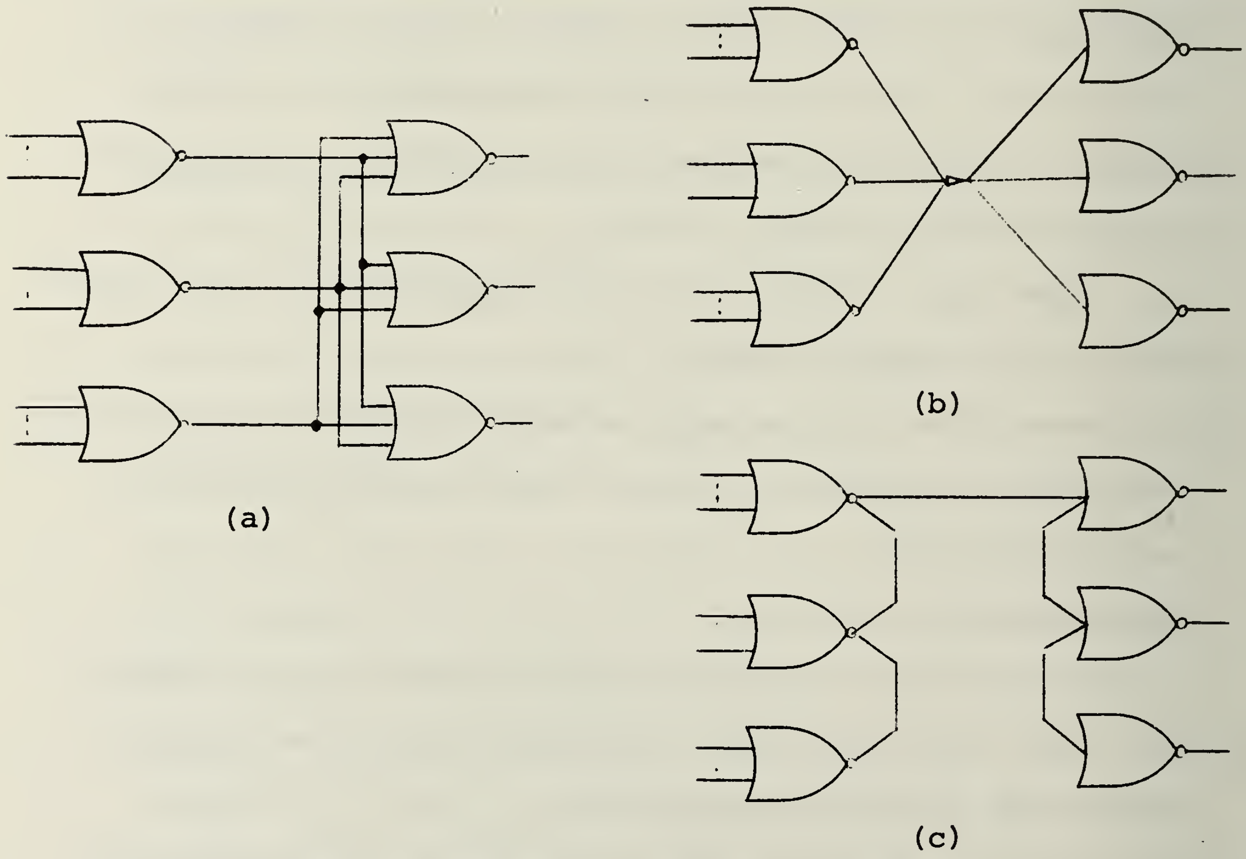


Figure 2.10--Examples of a Wired-OR

Generalizing this conversion, we can form a Wired-OR consisting of $k_1 + k_2$ connections generally for a set of connections from k_1 inputs going to k_2 outputs which require $k_1 \times k_2$ connections. Network (c) which consists of five connections is also feasible. For this connection configuration, we get a Wired-OR consisting of $k_1 + k_2 - 1$ generally. If we assume that this network (c) is located on a single IC chip, this counting method of the number of connections may be reasonable. It is very difficult to define the way of counting the number of connections for a general case. But counting as shown in the case (c) would be one of reasonable definitions, and this counting is used in this paper. Therefore, the number of connections required by a Wired-OR which has k_1 inputs and k_2 outputs is $k_1 + k_2 - 1$.

Next, let us discuss the algorithm in detail.

(C) At first in phase 3, Wired-ORs whose outputs are connected to only NOR-OR gates but are not used as network outputs are derived using networks obtained by Phase 2. If R_k denotes the number of gates which have a possibility to have at least one input from OR output in each network obtained by Phase 1, $2^{R_k} - 1$ possible new networks can be derived by Phase 2 from each original network obtained by Phase 1 by replacing or not replacing by OR-output-to-gate connections. Thus we have to check whether the total number of connections can be reduced by considering Wired-

ORs in each of all 2^{R_k} possible networks, i.e., $2^{R_k} - 1$ new networks and one original network. So we have to check R_k gates in each network to exhaust all Wired-ORs, and totally $2^{R_k} \times R_k$ gates in each original network obtained by Phase 1. But we can reduce this number of Theorem 3.

<u>Theorem 3</u> If R_k denotes the number of gates which have a possibility to have at least one input from OR output in each original network obtained by Phase 1, we need to check only $2^{R_k} - 1 + R_k$ gates for each original network to exhaust all Wired-ORs (instead of $2^{R_k} \times R_k$), by introducing a R_k bit down-counter. This R_k bit down-counter is used for deriving all 2^{R_k} possible networks in the following manner: Each count shows the status of each of 2^{R_k} networks such that each bit shows the status of each of all R_k gates; bit value "1" means that some inputs of the gate corresponding to the bit position are replaced by OR-output-to-gate connections and bit value "0" means that the inputs are not replaced by OR-output-to-gate connections but connected from the inputs of the gates as obtained by Phase 1; the count (0, 0, . . ., 0) denotes the original network.

<u>Proof</u> An example of this down-counter is shown in Figure 2.11. Initially all bits are set to "1." This means that some inputs of each gate of all R_k gates in the network corresponding to this count (1, 1, . . ., 1) are replaced by OR-ouput-to-gate connections. We need to check R_k

Gate #.	R_k	R_k−1	. . .	5	4	3	2	1
	1	0	. . .	0	1	1	0	1

When we have a count as shown above, for example, some inputs of gates 1, 3, 4 . . . and R_k are replaced by the respective OR-output-to-gate connections, and any inputs of gates 2, 5, . . . and R_k - 1 are not replaced by OR-output-to-gate connections but connected from the inputs of these gates as obtained by Phase 1.

Figure 2.11--A R_k bit down-counter

gates for this network to exhaust the possible Wired-ORs. The remaining $2^{R_k} - 1$ possible networks are derived by the counts of the counter by counting down one by one. The down-count has the basic property that exactly one bit is changed from "1" to "0" in each down-count, even though more than one bit may be changed from "0" to "1." For each of these $2^{R_k} - 1$ possible networks, we need to check only one gate whose corresponding bit of the R_k bit counter is changed from "1" to "0," to find whether a Wired-OR can reduce the number of connections when the inputs of the gate is not replaced by the corresponding OR-output-to-gate connection. But we do not need to check the gates whose corresponding bits are changed from "0" to "1." This is becuase we want to check whether Wired-ORs can reduce the number of connections in each network of $2^{R_k} - 1$ possible networks and because since gates whose corresponding bits are changed from "0" to "1" (this means that the original inputs to each of these gates will be replaced by OR-output-to-gate connections) were already checked in the initial network corresponding to the count (1, 1, . . ., 1) to find whether Wired-ORs can reduce the number of connections, there is no possibility to derive any new Wired-ORs which reduce the number of connections. Therefore corresponding to each count-down, we need to check only one gate for each network of the remaining $2^{R_k} - 1$ possible networks. This means checking $2^{R_k} - 1 \times R_k$ gates totally, i.e., R_k gates for the initial network and $2^{R_k} - 1$ gate for the succeeding $2^{R_k} - 1$ networks. Q.E.D.

An up-counter can also be used for this purpose.

(D) Secondarily, a Wired-OR whose output is used as the network output is derived by Theorem 2.

(E) Finally, networks which have the minimum number of NOR-OR gates first and the minimum number of connections second are derived among all networks for a given function, simply by counting the numbers of gates and connections in each network and comparing them.

3. LOGICAL DESIGN PROBLEMS BASED ON INTEGER PROGRAMMING FORMULATION

The Integer Programming (IP) problems dealt with in this paper are so-called 0 - 1 IP problems(9)(10) and formulated to calculate all networks with the minimum number (R_n) of NOR gates for a given function f. No fan-in and fan-out restrictions are assumed in this paper.

3.1 Representation of a NOR network with inequalities(3)(6)(13)

Assuming that the network synthesis problem has n external variables $X_1, X_2, \ldots, X_n$, all of the 2^n possible input vectors must be considered. For convenience each of the input vectors $\vec{X} = (X_1, X_2, \ldots, X_n)$ is numbered as $\vec{X}^{(j)}$ ($j = 1, 2, 3, \ldots, r = 2^n$) from $\vec{X}^{(1)} = (0, 0, \ldots, 0)$ through $\vec{X}^{(r)}$ $(1, 1, \ldots, 1)$. Let W_i^k represent the connection from external variable X_i to gate k. If $W_i^k = 1$, the connection exists. If $W_i^k = 0$, the connection does not exist. Let the connection from gate i to gate k ($i \neq k$) be denoted by α_{ik}. The connection exists if $\alpha_{ik} = 1$ and does not exist if $\alpha_{ik} = 0$. Let $P_i^{(j)}$ denote the output value of gate i for input vector $\vec{X}^{(j)}$. Finally, let

$$\beta_{ik}^{(j)} = \alpha_{ik} P_i^{(j)} \tag{3.1}$$

(i) Inequalities for a network

Using the preceeding definitions, inequalities characterizing a network with R_n NOR gates are as follows:

$$\sum_{l=1}^{n} W_l^k X_l^{(j)} + \sum_{\substack{i=1\\i\neq k}}^{R_n} \beta_{ik}^{(j)} \geq 1 - UP_k^{(j)} \quad (3.2)$$

$$-\sum_{l=1}^{n} W_l^k X_l^{(j)} - \sum_{\substack{i=1\\i\neq k}}^{R_n} \beta_{ik}^{(j)} \geq - U (1 - P_k^{(j)}) \quad (3.3)$$

where $j = 1, 2, 3, . . ., 4 = 2^n$ and $k = 1, 2, 3, . . ., R_n$. Here U is a positive number large enough so that inequality (3.2) is non-restrictive if $P_k^{(j)} = 1$ and inequality (3.3) is non-restrictive if $P_k^{(j)} = 0$. We assume that gate 1 is the network output gate, so

$$P_1^{(j)} = f_R (X^{(j)}) \quad (3.4)$$

Inequalities for possible combinations among all the gates are necessary for the entire network. Thus all connections but inequalities for preventing loops are included. This is called "all-interconnection formulation."(15)

(ii) Inequalities for $\beta_{ik}^{(j)} = \alpha_{ik} P_i^{(j)}$

As can be easily seen, the non-linear equality (3.1) can be replaced by the following linear inequalities:

$$-P_i^{(j)} - \alpha_{ik} + \beta_{ik}^{(j)} \geq 1 \quad (3.5)$$

$$P_i^{(j)} + \alpha_{ik} - 2\beta_{ik}^{(j)} \geq 0 \quad (3.6)$$

where $i = 1, 2, . . ., R_n$, $k = 1, 2, . . ., R_n$ $(i \neq k)$ and $j = 1, 2, . . ., r = 2^n$.

(iii) Restrictions on inputs and outputs

Each gate except the network output gate must have at least one input from the external variables or from other gates, becuase we do not need any gates which have no inputs. Thus

$$\sum_{l=1}^{n} W_l^k + \sum_{\substack{i=1 \\ i \neq k}}^{R_n} \alpha_{ik} \geq 1 \qquad (3.7)$$

where k = 2, 3, . . ., R_n.

All gates except the network output gate connect to at least one other gate.

Thus

$$\sum_{\substack{j=1 \\ j \neq k}}^{R_n} \alpha_{kj} \geq 1 \qquad (3.8)$$

where k = 2, 3, 4, . . ., R_n

The following three inequalities are powerful for speeding up the computation.

(iv) Triangular condition.

Theorem 4(2)(3)(13) Suppose three gates are connected as shown in Figure 3.1 where the only output from gate j is α_{jk}. If $\alpha_{ij} = \alpha_{jk} = \alpha_{ik} = 1$, the network shown in Figure 3.1 is not optimal.

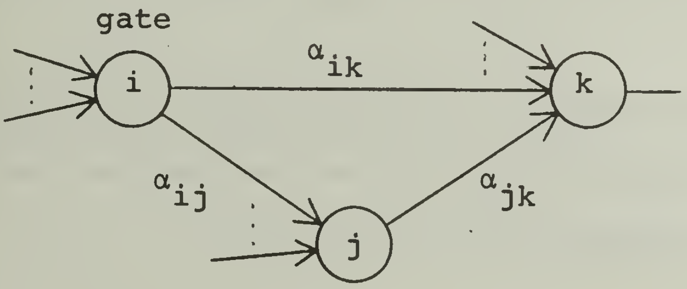


Figure 3.1--Triangular connections

By this theorem, at least one of α_{ij}, α_{jk} and α_{ik} is 0. Thus

$$\alpha_{ij} + \alpha_{jk} + \alpha_{ik} \leq 2 + \sum_{\substack{l=1 \\ l \neq i,\, j,\, k}}^{R_n} \alpha_{jl} \qquad (3.9)$$

where i, j, k = 1, 2, . . ., R_n and i ≠ j ≠ k ≠ i.

This property is still true even if gate i is replaced by external variable X_i. Thus

$$W_i^j + W_i^k + \alpha_{jk} \geq 2 + \sum_{\substack{l=1 \\ l \neq j,\, k}}^{R_n} \alpha_{jl} \qquad (3.10)$$

where j, k = 1, 2, . . ., R_n, i = 1, 2, . . ., n and k≠j.

(v) Restrictions on the inputs of the network output gate.

<u>Theorem 5</u>(2)(3)(13) All gates which are connected to the network output gate (gate 1) are not connected to any other gate.

The property stated in this theorem is expressed

$$-\sum_{\substack{k=2 \\ k \neq i}}^{R_n} \alpha_{ik} \geq U\ (\ \alpha_{il} - 1) \qquad (3.11)$$

U is a positive integer large enough so that the inequality becomes non-restrictive if $\alpha_{il} = 0$.

3.2 Procedure

A synthesis approach for all networks which have a minimum number of NOR gates without minimizing the number of connections is as follows(6):

(i) The initial value of R_n is set to 1; $R_n = 1$.

(ii) Formulate an IP problem described in section 3.1 to obtain the networks for f using R_n NOR gates. The objective function is not required.

(iii) If there exists no solution, set $R_n + 1 \rightarrow R_n$ and repeat (ii) and (iii) until solutions are obtained.

By this step, all networks which have the minimum number of NOR gates but where the number of connections is not necessarily minimized are obtained for a given function f.

3.3 Notations

R_n: A number of NOR gates assumed in a network.

f: A function to be realized.

$X_1, X_2, \ldots, X_n$: n external variables.

$\vec{X}^{(j)} = (X_1^{(j)}, X_2^{(j)}, \ldots, X_n^{(j)})$: j-th input vector, where $X_i^{(j)} =$ 0 or 1 and $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, r = 2^n$.

α_{ik}: The connection from gate i to gate k exists if $\alpha_{ik} = 1$ and does not exist if $\alpha_{ik} = 0$.

W_i^k: The connection from external variable X_i to gate k exists if $W_i^k = 1$ and does not exist if $W_i^k = 0$.

$P_k^{(j)}$: The output value of gate k for the j-th input vector $\vec{X}^{(j)}$.

$\beta_{ik}^{(j)} = P_i^{(j)} \alpha_{ik}$: The value of the input to gate k from gate i for the j-th input vector $\vec{X}^{(j)}$.

U: A sufficiently large positive number.

Table 3.1--Notation table

4. ALGORITHM FOR THE DERIVATION OF OPTIMAL NETWORKS WITH NOR-OR GATES AND WIRED-OR LOGIC

The algorithm consists of three phases--solution of the logical design problems based on the IP formulation, derivation of OR-output-to-gate connections and derivation of Wired-ORs.

The flow chart is shown in Figure 4.1.

4.1 Phase 1: Solution of the logical design problems based on the IP formulation.

Since the algorithm for this is described in section 3.2, it is omitted here.

4.2 Phase 2: Derivation of OR-output-to-gate connections.

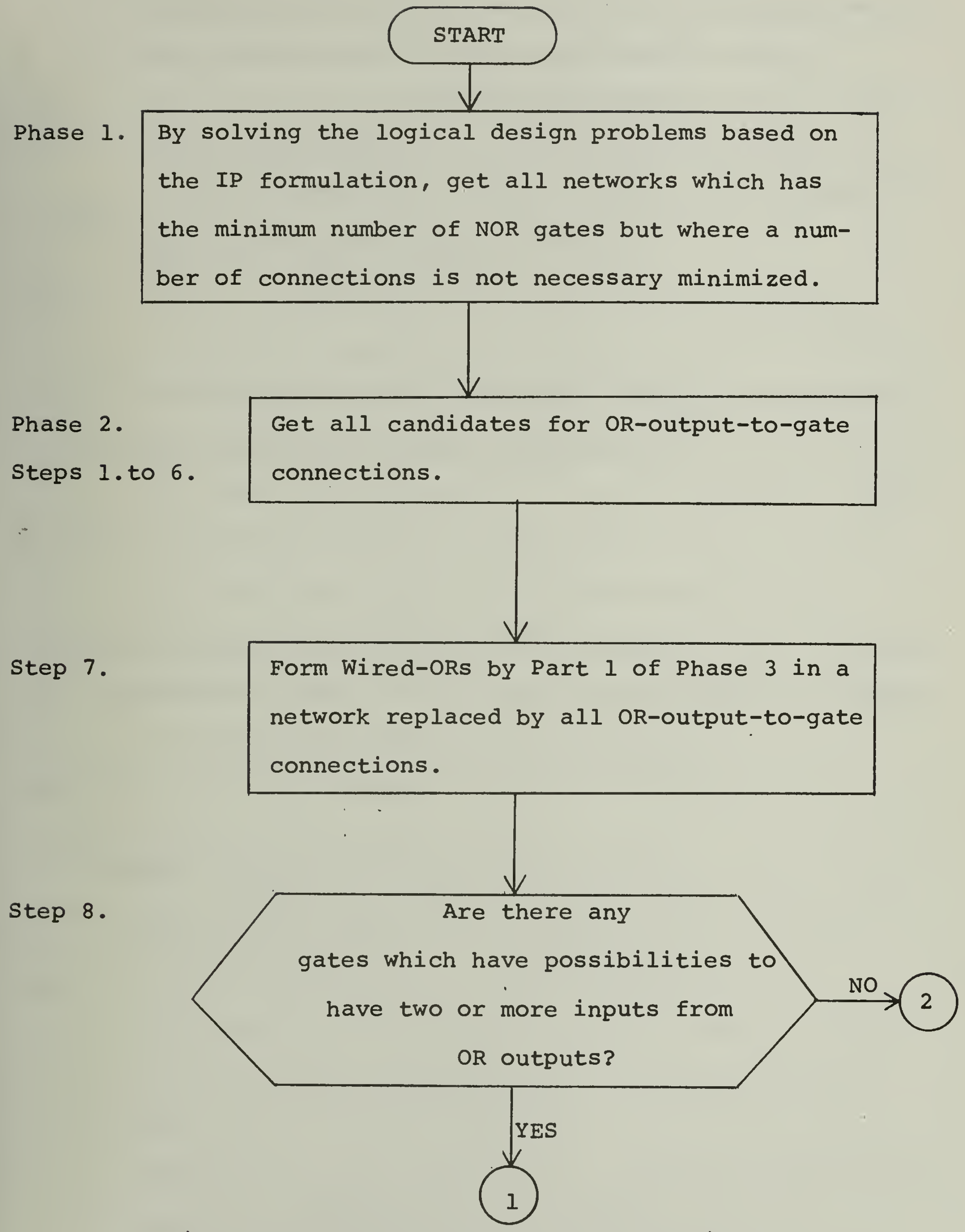
Step 1. Choose a network among those derived by Phase 1.

Step 2. Pick up a gate k which satisfied (i) of restriction 1, where $1 \leq k \leq R_n$.

Step 3. Get all seccessor gates of gate k.

Step 4. Choose a gate as gate j which satisfies (ii) and (iii) of restriction 1 and where the set of all inputs of gate j is identical to a subset of inputs of gate k, where $2 \leq j \leq R_n$. Check whether each gate in the network can be a candidate for gate j.

Step 5. Classify a gate k into set S if this gate has a possibility to have only one input from OR output, and



START
Phase 1.
By solving the logical design problems based on the IP formulation, get all networks which has the minimum number of NOR gates but where a number of connections is not necessary minimized.
Phase 2.
Steps 1.to 6.
Get all candidates for OR-output-to-gate connections.
Step 7.
Form Wired-ORs by Part 1 of Phase 3 in a network replaced by all OR-output-to-gate connections.
Step 8.
Are there any gates which have possibilities to have two or more inputs from OR outputs?
NO
2
YES
1

Figure 4.1--A flow chart of the algorithm

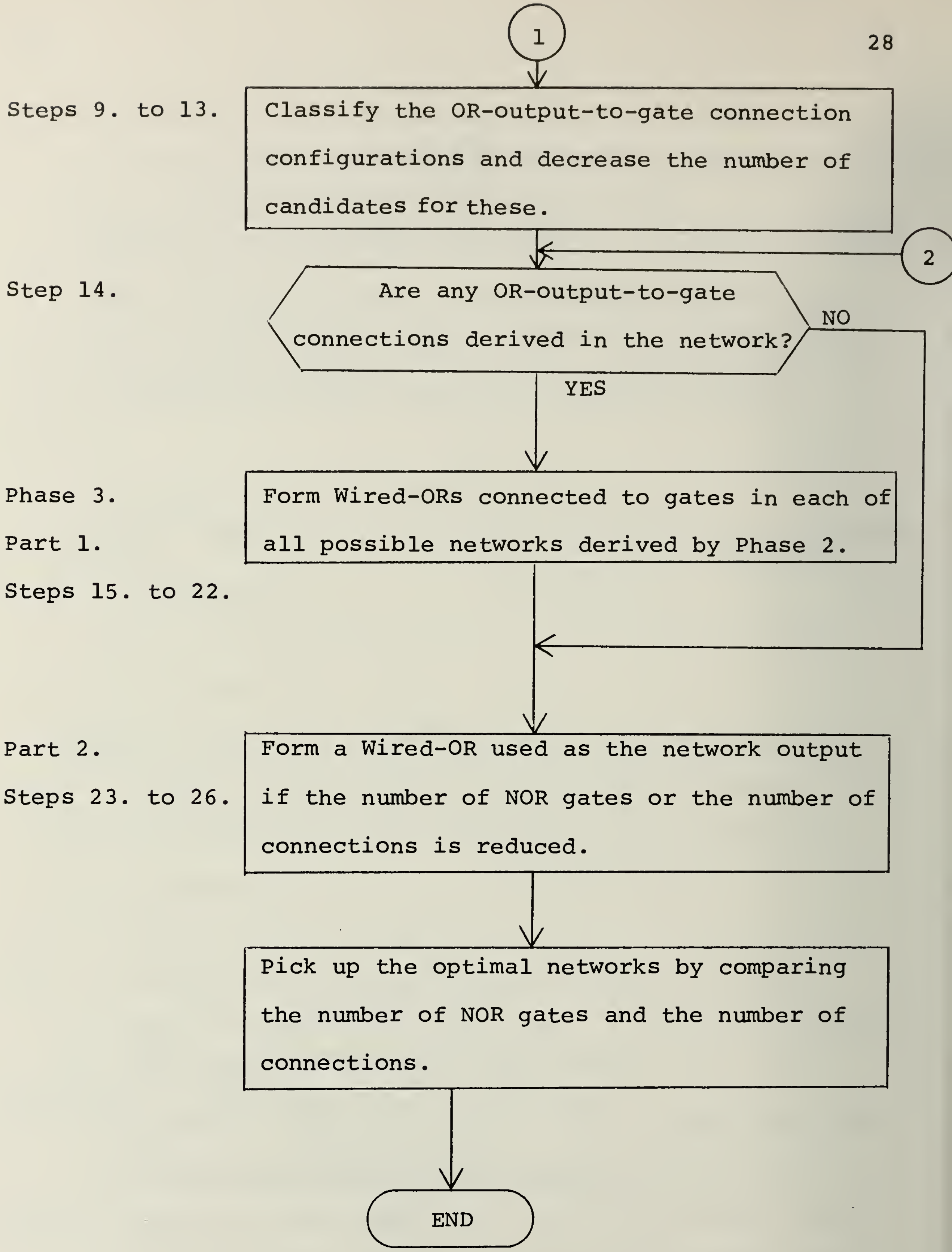

1
Steps 9. to 13.
Classify the OR-output-to-gate connection configurations and decrease the number of candidates for these.
2
Step 14.
Are any OR-output-to-gate connections derived in the network?
NO
YES
Phase 3.
Part 1.
Steps 15. to 22.
Form Wired-ORs connected to gates in each of all possible networks derived by Phase 2.
Part 2.
Steps 23. to 26.
Form a Wired-OR used as the network output if the number of NOR gates or the number of connections is reduced.
Pick up the optimal networks by comparing the number of NOR gates and the number of connections.
END

Figure 4.1--(Continued)
A flow chart of the algorithm

classify it into set M if this gate has possibilities to have two or more inputs from OR outputs.

Step 6. Repeating Step 2 through Step 5, check whether each gate in the network can be a candidate for gate k. Let R_k be the number of gates which have at least one input from OR output.

Step 7. Derive all the Wired-ORs in the network by part 1 of Phase 3 replacing the original connections by the corresponding OR-output-to-gate connections which are obtained previously as candidates. Let WG be the number of Wired-ORs.

Step 8. Go to Step 14 if there is no gate in set M. Go to the next step if there are gates in set M.

Step 9. Let R_t be the number of inputs from OR outputs to a gate t in set M. Make a R_t bit down-counter and set all the bits to "1".

Step 10. In the corresponding connection configuration of gate t,

(i) replace by OR-output-to-gate connections to inputs of gates whose corresponding bits in the down-counter are "1",

(ii) but do not replace by OR-output-to-gate connections to inputs of gates whose corresponding bits are "0".

(Theorem 3 is not applied here. It will be applied in Phase 3, Part 1.)

Step 11. Repeat Step 9 through Step 10 for all $2^{Ro} - 1$ possible subnetworks by counting down a Ro bit down-counter by one. Classify the connection configuration to a gate in set M by the number of input connections.

Step 12. Choose connection configurations which satisfy restriction 2.

Step 13. Repeat Step 9 through Step 12 for every gate in set M.

Step 14. If $R_k = 0$, go to Part 2 of Phase 3. (If WG = 0, the network by Step 1 can not be minimized by Part 1 and if WG $\neq$ 0, the network by Step 7 can not be minimized any more by Part 1, where WG is defined in Step 7.) If $R_k \neq 0$, go to Part 1 of Phase 3.

4.3 Phase 3: Derivation of Wired-ORs

(i) Part 1. Derivation of Wired-ORs whose outputs are connected to gates.

Step 15. Make a R_k bit down-counter in which each bit represents each of all the R_k gates. Set all bits to "1."

Step 16. Make a network by replacing the original connections by OR-output-to-gate connections to inputs of gates in set S and by OR-output-to-gate connections in Class 1 to inputs of gates in set M. Let the current count N of a R_k bit down-counter represent a network N.

Step 17. For each input 1_i of gate 1, find all gates to which this input is connected. Denote these gates with $(l_{i1}, l_{i2}, \ldots, l_{imi})$.

Step 18. Thus we have n gate sets $(l_{11}, l_{12}, \ldots, l_{1m1})$, $(l_{21}, l_{22}, \ldots, l_{2m2}), \ldots, (l_{n1}, l_{n2}, \ldots, l_{nmn})$ corresponding to inputs $l_1, l_2, \ldots, l_n$ of gate l. Check whether there are identical sets among these gate sets applying Restriction 3. If there are any, store

(i) those inputs as candidates for inputs of Wired-ORs,

(ii) the corresponding gate sets as candidates for outputs of Wired-ORs,

(iii) the number of connections as R_{mc} after replacing the original connections by all Wired-ORs, where this R_{mc} is initialized to be large positive number,

(iv) the network number N,

(v) and the network itself.

Repeat Steps 17 and 18 for all l, where $2 \leq l \leq R_n$.

Step 19. Count down the R_k bit counter by one, and generate a network corresponding to the new count.

Step 20. Check the possibility to form Wired-ORs for only the gate whose corresponding bit of the counter changes from "1" to "0", by Steps 17 and 18. If there are any new Wired-ORs, compare the number of connections in the entire network with R_{mc}. Store (i) through

(v) of Step 18 only when the number of connections of the new network is smaller than R_{mc}.

Repeat Steps 19 and 20 until each bit of the counter becomes "0".

Step 21. Set N into the R_k bit counter. Corresponding to N, pick up a gate m which is in set M, and whose corresponding bit of the R_k bit counter is "1." If there is none, go to Part 2.

Step 22. Change OR-output-to-gate connections for inputs of gate m from class i to class i + 1, where i = 1, 2, 3, . . ., and check the possibility to form Wired-Ors by the procedure described in Steps 17 and 18 for gate m. If a network which has a fewer connections than R_{mc} is obtained, store the number of connections. Repeat for all classes of connection configurations of gate m, and repeat for each gate as gate m.

(ii) Part 2. Derivation of a Wired-OR whose output is used as the network output.

Step 23. Repeat Step 2 through Step 22 for all original networks for a function f, and pick up the network which has the minimum number of connections.

Step 24. Repeat Step 2 through Step 23 for all original networks for a given function obtained by Phase 1.

Step 25. Let R_n and R_{cn} be the minimum numbers of NOR-OR gates in the networks among the networks obtained by Step

24 for f and $\overline{f}$, respectively. If $R_{cn} \leq R_n$, (this means $R_n = R_{cn} + 1$ or $R_n = R_{cn}$ by Theorem 2) change gate 1 of the network for $\overline{f}$ to a Wired-OR. Then this network has become an optimal network for f. If $R_{cn} > R_n$, go to the next step.

Step 26. If the number of connections in the network for $\overline{f}$ is larger than that for f, this network for f is optimal. Otherwise, change gate 1 of the network for $\overline{f}$ to a Wired-OR. Then this has become an optimal network for f.

When we repeat all steps from 1 to 26 for all functions to be solved, we will have a catalog.

5. COMPUTATIONAL RESULTS BY THE ALGORITHM

5.1 Program package

This program package is an implementation of Phase 2 and Part 1 of Phase 3 of the algorithm, and can treat up through four variable functions which require 15 NOR-OR gates. Phase 1 is accomplished by the integer program package implemented in the past (Developed by Muroga et. al., and modulated by Y. Kembayashi), and Part 2 of Phase 3 is accomplished by hands. This program package consists of one main program and seven subroutines written in FORTRAN IV. The number of statements for each program is shown in Table 5.1, and the details of flow charts and programs are shown in Appendix C and D, respectively.

Name	Function	Number of Statements
MAIN	Main program	338
CHKWRD	Check whether a new Wired-OR satisfies Restriction 3	73
CONCT	Count connections in the entire network	15
DOWNCT	Count down a counter by one	13
MAKECT	Generate or change networks	16
SEQNS	Sort by the number of gates	17
SECSSR	Obtain all successor gates of a gate	23
WRDOR	Derive all Wired-ORs in the network	94

Table 5.1--Main program and seven subroutines in the program package

5.2 Optimal networks

The algorithm was applied to all networks for all three variable functions and to all networks for all four variable functions which can be implemented with five or less NOR gates, where all these networks were obtained by Phase 1. All functions are classified into equivalence classes with respect to permutation of variables. According to this classification, there are 77 representative functions for functions of three or less variables except trivial functions (i.e., 0, 1, and x_i) and 312 representative functions for functions of exactly four variables which can be implemented with five or less NOR gates (it was proved by Ikeno et. al.[14] that 312 representative functions can be implemented with five or less NOR gates). For original networks obtained by Phase 1, the numbers of representative functions and numbers of networks are shown in Table 5.2 according to the number of NOR gates in a network.

	Number of NOR gates in each network						
	1	2	3	4	5	6	7
Number of representative functions	3	5	8	17	23	15	6
Number of networks by Phase 1	3	5	12	35	94	235	189
Number of OR-output-to-gate connections by Phase 2	0	0	0	0	1	10	15
Number of Wired-ORs by Part 1 of Phase 3	0	0	0	2	9	19	52
Number of Wired-ORs by Part 2 of Phase 3	0	2	4	11	21	14	6

Table 5.2 (a)--Statistics on functions of three variables at each phase of the algorithm

Numbers of all OR-output-to-gate connections and numbers of all Wired-ORs in networks obtained by Phase 2 and by Part 1 of Phase 3 (by the program package), and numbers of Wired-ORs in optimal networks obtained by Part 2 of Phase 3 are also shown in the same table. (e.g., 52 Wired-ORs are found throughout 189 networks of 7 gates.)

The number of Wired-ORs used as the network output of networks which can be implemented with five NOR gates and also the number of representative functions are not found in the cases labeled with * in Tables 5.2 (b) and 5.3 (b) for the following reason. For functions of four variables, no network

	Number of NOR gates in each network				
	1	2	3	4	5
Number of representative functions	1	4	13	60	234
Number of networks by Phase 1	1	4	20	135	892
Number of OR output-to-gate connections by Phase 2	0	0	0	1	23
Number of Wired-Ors by Part 1 of Phase 3	0	0	2	27	221
Number of Wired-ORs by Part 2 of Phase 3	0	1	5	37	*

Table 5.2 (b)--Statistics on functions of four variables at each phase of the algorithm

which can be implemented with six NOR gates was obtained by Phase 1. If they were obtained, the number of NOR gates in a network could be reduced by Theorem 2.

For optimal networks with NOR-OR gates and Wired-ORs which are derived by the algorithm, numbers of representative functions are shown in Table 5.3 classified according to the number of NOR-OR gates in each optimal network. For functions

	Number of NOR-OR gates in each network							
	1	2	3	4	5	6	7	Total
Number of representative functions	6	7	17	24	15	7	0	75

Table 5.3 (a)--Optimal networks with NOR-OR gates and Wired-ORs for functions of three or less variables.

	Number of NOR-OR gates in each network				
	1	2	3	4	5
Number of representative function	4	13	60	234	*

Table 5.3 (b)--Optimal networks with NOR-OR gates and Wired-ORs for functions of four variables

of three or less variables, the numbers of NOR gates of two functions (i.e., $X_2 \vee X_3$ and $X_1 \vee X_2 \vee X_3$) are reduced to zero.

This means that networks for these functions are implemented without NOR-OR gate but with single Wired-ORs. For functions of four variables, the number of NOR gates of one function (i.e., $X_1 \vee X_2 \vee X_3 \vee X_4$) is reduced from two to zero. Thus these functions are not counted in Table 5.3.

5.3 Major conputational results

By combining exhaustive methods with the IP-formulation, computer time is reduced greatly compared with the approach based on only the IP-formulation. Phase 2 and Part 1 of Phase 3, however, are exhaustive methods, taking only 200 seconds for all switching functions of three or less variables and 130 seconds for all switching functions of four variables which can be implemented with five or less NOR gates.

Compared with Hellerman's catalog(1) (The results by integer programming approach(15) are identical to Hellerman's in the case of three variable functions) and with the results of the IP-formulation(12) for all switching functions of three or less variables and for all switching functions for four variables and for all switching functions for four variables which can be implemented with four or less NOR gates, this algorithm reduced 88 connections (by Theorem 1) and 74 NOR gates (by Theorem 2) throughout 59 functions of 77 representative functions of three or less variables and 74 connections and 28 NOR gates throughout 45 functions of 78 representative functions of four variables. About 16% of connections and 26% of NOR gates on the average for 77 representative functions of three or less variables, and 11% of connections and 10% of

NOR gates on the average of 78 representative functions of four variables are reduced by replacing by OR output-to-gate connections and by Wired-ORs.

The numbers of representative functions which are improved by the algorithm are shown in Table 5.4. Whenever an optimal NOR network is improved by the algorithm, OR outputs and Wired-ORs are simultaneously incorporated or Wired-ORs without OR outputs are incorporated.

	No improvement	Improved by OR alone	Improved by Wired-ORs alone	Improved by both
Number of representative functions of three variables	18	0	53	6
Number of representative functions of four variables	33	0	44	1

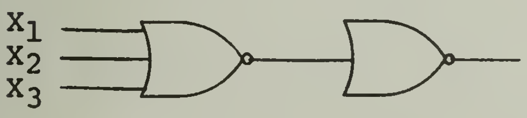
Table 5.4--The numbers of representative functions improved by the algorithm

5.4 Examples of optimal networks

Three examples of optimal networks are shown here comparing with optimal networks of NOR gates only derived by Hellerman's catalog.(1)

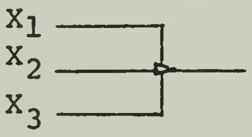
(i) $X_1 \vee X_2 \vee X_3$

Hellerman's catalog



2 NOR gates, 4 connections

This algorithm

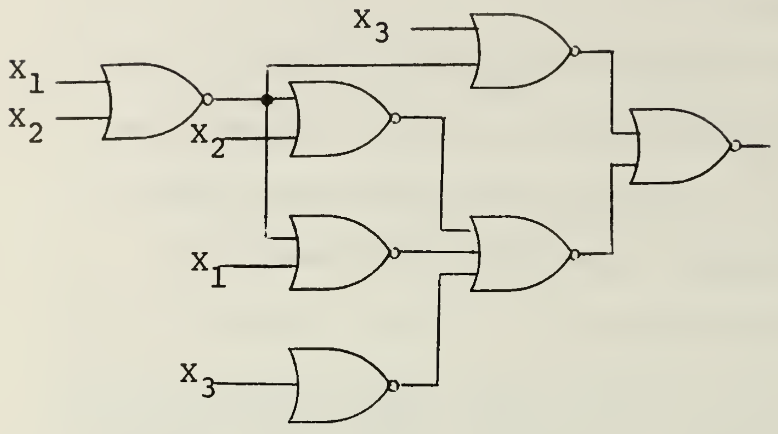


2 connections

Two NOR gates and two connections are reduced.

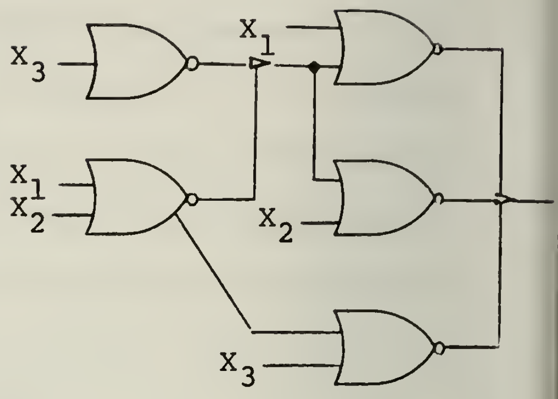
(ii) $x_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$

Hellerman's catalog



7 NOR gates, 14 connections

This algorithm

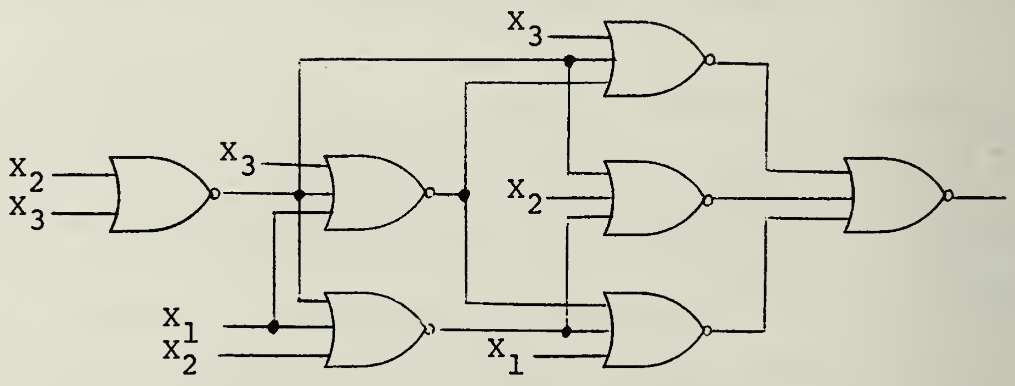


5 NOR gates, 12 connections

Two NOR gates and two connections are reduced.

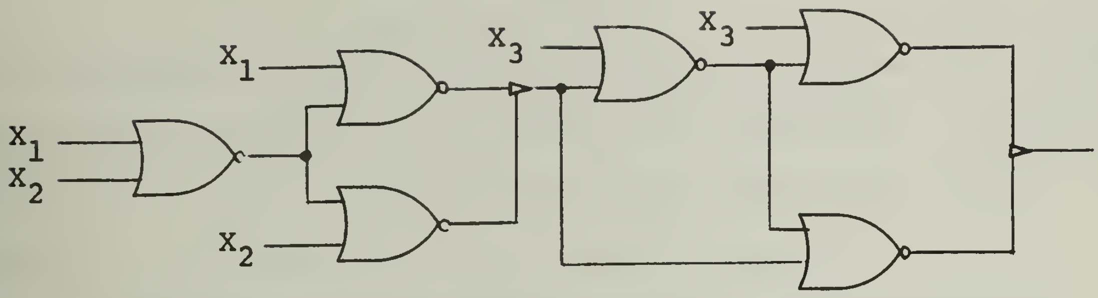
(iii) Parity function $x_1x_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3$

Hellerman's catalog



7 NOR gates, 20 connections

This algorithm



6 NOR gates, 14 connections

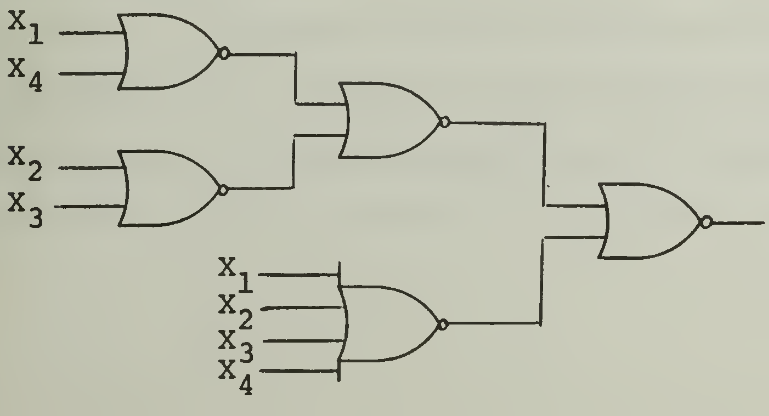
One NOR gate and six connections are reduced.

All optimal networks whose numbers of NOR gates or numbers of connections are reduced by Phases 2 and 3 of this algorithm are shown in Appendix A-1 and B-1 for all switching functions of three or less variables.

Hellerman's catalog has no network for functions of four variables, but networks which consists of only NOR gates have been obtained by the logical design based on the IP-formulation.(12) Let us show examples of optimal networks for functions of four variables comparing with the networks derived by the logical design based on the IP-formulation.

(i) $X_1\overline{X}_2\overline{X}_3 \vee \overline{X}_1X_2\overline{X}_4 \vee \overline{X}_1X_3\overline{X}_4 \vee \overline{X}_2\overline{X}_3X_4$

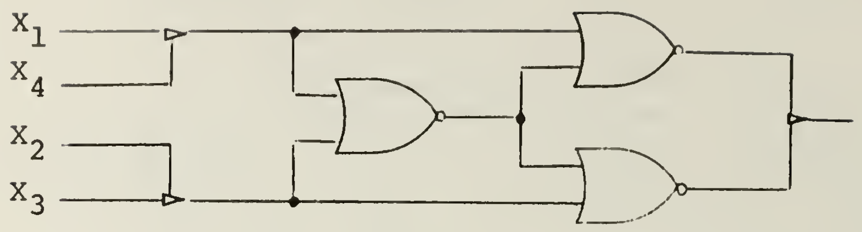
IP-formulation



5 NOR gates and

12 connections

This algorithm



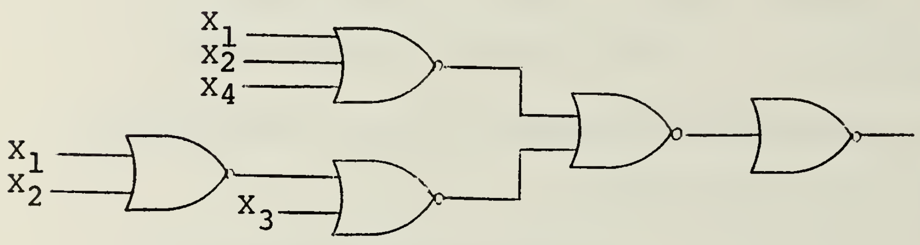
3 NOR gates and

9 connections

Two NOR gates and three connections are reduced.

(ii) $X_1\bar{X}_3 \vee X_2\bar{X}_3 \vee \bar{X}_1\bar{X}_2\bar{X}_4$

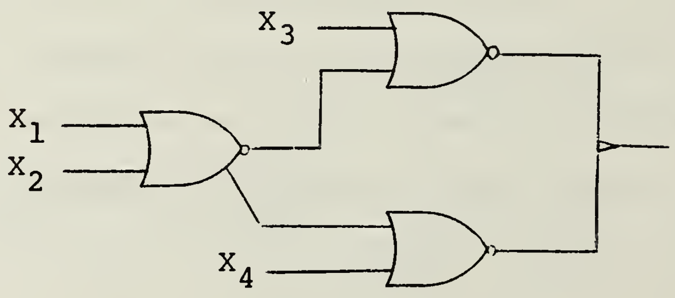
IP-formulation



5 NOR gates and

10 connections

This algorithm



3 NOR gates and

7 connections

Two NOR gates and three connections are reduced.

All optimal networks whose numbers of NOR gates or numbers of connections are reduced by Phases 2 and 3, and which can be implemented with three or less NOR-OR gates are shown in Appendix A-2, and B-2.

6. CONCLUSION

Optimal networks consisting of NOR-OR gates and Wired-ORs under the assumption that only non-complemented variables are available as the network inputs have been found for all switching functions of three or less variables and for all switching functions of four variables which can be implemented with three or less NOR-OR gates, defining the optimality as the minimization of the number of gates first, and the number of connections next. The reason why we defined the optimality in this manner is that the cost of connections is considered to be very small compared with the cost of implementation of NOR-OR gates. But if the cost of connections becomes comparable to or is sometimes higher than that of NOR-OR gates, we have to consider a different definition of the optimality.

Also a Wired-OR is defined to have $k_1 + k_2 - 1$ connections. If we define it to be $k_1 + k_2$ as in the case (b) in Figure 2.10 (this also may be a fiarly reasonable definition), all optimal networks which have at least one Wired-OR derived by this algorithm are still optimal networks. And then the original networks which we had before replacing the original connections by these Wired-ORs are also optimal networks for all the functions except only two functions $\overline{X}_1 X_2 \overline{X}_3 \overline{X}_4 \vee X_1 \overline{X}_2 \overline{X}_3 \overline{X}_4$ and $(X_1 \vee X_2 \vee X_3) \overline{X}_4 \vee \overline{X}_1 \overline{X}_2 \overline{X}_3 X_4$ for the following reason:

Each of all the Wired-ORs in these networks except those for the above two functions has exactly two inputs and two outputs, in other words $k_1 = k_2 = 2$, and these Wired-ORs do not reduce the number of connections, because if we do not use this Wired-OR, $k_1 \times k_2 = 4$ connections are required (notice that the number of connections for the Wired-OR is $k_1 + k_2 = 4$). Consequently, each function except the above two functions has latter networks in each of which the number of connections is equal to that of the former networks. Thus each function except the above two functions has one extra optimal network without Wired-ORs, corresponding to each optimal network with Wired-ORs for the function.

However each of the two functions $\overline{X}_1X_2\overline{X}_3\overline{X}_4 \vee X_1\overline{X}_2\overline{X}_3\overline{X}_4$ and $(X_1 \vee X_2 \vee X_3)\ \ \overline{X}_4 \vee \overline{X}_1\overline{X}_2\overline{X}_3X_4$ has one optimal network which contains one Wired-OR with two inputs and three outputs, under the counting of $k_1 + k_2 - 1$. The Wired-OR reduces the number of connections. Thus the corresponding network without the Wired-OR is not an optimal network under the counting of $k_1 + k_2$.

The networks for functions of three or less variables which are obtained by Phase 1 are also derived by the branch-and-bound method to make sure whether all networks are correctly derived by Phase 1.

LIST OF REFERENCES

(1) L. Hellerman, "A catalog of three Variable OR-Invert and AND-Invert Logical Circuits," IEEE Trans. Electron. Comput., Vol. EC-12, pp. 198-223, June 1963.

(2) S. Muroga, "Logical design of optimal digital networks by integer programming," in Advances in Information Systems Science, J.T. Ton Ed. New York Plenum, 1970, Vol, 3, Ch. 5, pp. 283-348.

(3) _____, "Threshold Logic and its Applications," New York Wiley-Interscience, 1971, Ch. 14.

(4) _____ & T. Ibaraki, "Design of optimal switching network by integer programming," IEEE Trans. Comput., Vol C-21, No. 6, pp. 573-582, June 1972.

(5) C.R. Bough, C. S. Chandersekaram, R.S. Swee & S. Muroga, "Optimal network of NOR-OR gates for functions of three variables," IEEE Trans. Comput., Vol. C-21, No. 2, PP. 153-160, Feb. 1972.

(6) Y. Kambayashi & S. Muroga, "Properties of Wired Logic," To be published.

(7) S.F. Gimpel, "The minimization of TANT networks," IEEE Trans. Electron, Comput. Vol. EC-16, pp. 18-38, Feb. 1967.

(8) S. Muroga & T. Ibaraki, "Logical Design of an optimum network by integer linear programming--Part I," Dept. Comput. Sci., Univ. Illinois, Urbana, Rep. 264, July 1968.

(9) _____, "_____--Part II," Dept. Comput. Sci., Univ. Illinois, Urbana, Rep. 289, Dec. 1968.

(10) T. Ibaraki, T.K. Liu, C. R. Baugh and S. Muroga, "Implicit enumeration program for zero-one integer programming," Int. J. of Comp. and Inf. Sci., Vol. 1, No. 1, pp. 75-92, March, 1972.

(11) T.K. Liu, "A code for zero-one integer linear programming by implicit enumeration," M.S. thesis, Dept. Comput. Sci., Univ. Illinois, Rep. 302, 1968.

(12) J.N. Culliney, "On the synthesis by integer programming of optimal NOR gate networks for four variable switching functions," M.S. thesis, Dept. Comput. Sci., Univ. Ill., Rep. 480, 1971.

(13) K.R. Holhulin, "A code for designing optimal networks by implicit enumeration using the all-interconnection inequality formulation." To be published.

(14) N. Ikeno, A. Hashimoto & K. Naito, "A Table of Four-Variable Minimal NAND Circuits," Electrical Communication Lab. Tech. J., Extra Issue No. 26, Electrical Communication Laboratory, Nippon Telegraph and Telephone Public Corporation, Tokyo, Japan, 1968 (in Japanese).

(15) C.R. Baugh, T. Ibaraki, T.K. Liu & S. Muroga, "Optimum network design using NOR and NOR-AND gates by integer programming," Dept. Comput, Sci., Univ. Illinois, Report No. 293, Jan. 10, 1969.

APPENDIX A

Optimal Networks With NOR-OR Gates And Wired-ORs

All optimal networks whose numbers of NOR gates or numbers of connections are reduced by the algorithm are shown here. The networks for functions of three or less variables are shown in A-1, and the networks for functions of four variables are shown in A-2.

Function number FCT is defined as follows:

(i) Functions of three or less variables.

External Variables									
	X_1	0	0	0	0	1	1	1	1
	X_2	0	0	1	1	0	0	1	1
	X_3	0	1	0	1	0	1	0	1
		f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7

FCT = $f_7f_6\ f_5f_4f_3\ f_2f_1f_0$ in octal.

e.g., for $X_1 \vee X_2X_3$; f_3, f_4, f_5, f_6 and f_7 =1

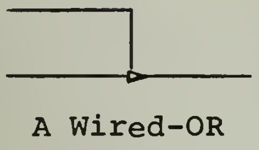
FCT = $(11111000)_2 = (270)_8$

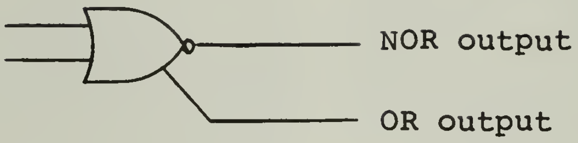
(ii) Functions of four variables.

External Variables																	
	X_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	X_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	X_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}

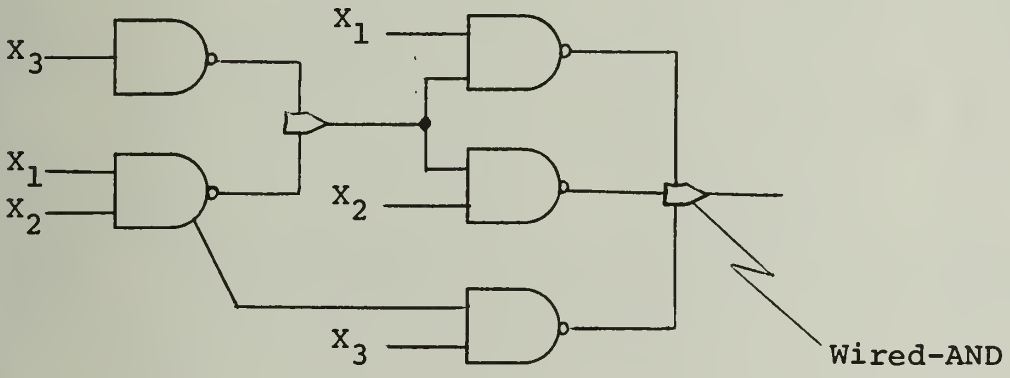
FCT = $f_0f_1f_2f_3\ f_4f_5f_6f_7\ f_8f_9f_{10}f_{11}\ f_{12}f_{13}f_{14}f_{15}$ in hexa-decimal.

Notations used in this paper are as follows:



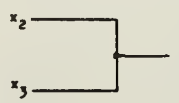


These tables show all networks with NOR-OR gates and Wired-ORs. From these tables, optimal networks with NAND-AND gates and Wired-AND can be easily obtained by finding the function number FCT for NAND for a given function f in the tables and then by changing NOR-OR gates and Wired-ORs to NAND-AND gates and Wired-ANDs, respectively. (e.g., the network with NOR-OR gates and Wired-ORs for function $\bar{x}_1\bar{x}_2\bar{x}_3$ V $\bar{x}_1x_2x_3$ V $x_1\bar{x}_2x_3$ (No. 42) is changed to the network with NAND-AND gates and Wired-ANDs for function $\bar{x}_1\bar{x}_2$ V $\bar{x}_1x_3$ V $\bar{x}_2x_3$ V $x_1x_2\bar{x}_3$ as shown in this figure.)



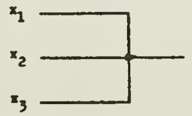
A-1 Functions of three or less variables

	FCT	EXPRESSION
NOR	356	$x_2 \vee x_3$
NAND	210	x_2x_3



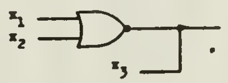
Network no. 1

	FCT	EXPRESSION
NOR	376	$x_1 \vee x_2 \vee x_3$
NAND	200	$x_1x_2x_3$



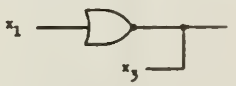
Network no. 2

	FCT	EXPRESSION
NOR	253	$\bar{x}_1\bar{x}_2 \vee x_3$
NAND	52	$(\bar{x}_1 \vee \bar{x}_2)x_3$



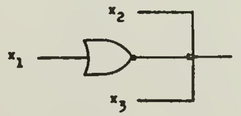
Network no. 3

	FCT	EXPRESSION
NOR	257	$\bar{x}_1 \vee x_3$
NAND	12	$\bar{x}_1x_3$



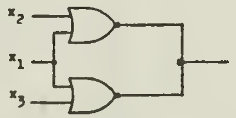
Network no. 4

NOR	357	$\bar{x}_1 \vee x_2 \vee x_3$
NAND	10	$\bar{x}_1x_2x_3$



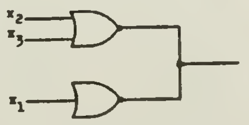
No. 5

NOR	7	$\bar{x}_1(\bar{x}_2 \vee \bar{x}_3)$
NAND	37	$\bar{x}_1 \vee \bar{x}_2\bar{x}_3$



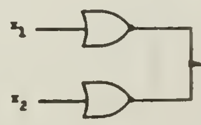
No. 6

NOR	37	$\bar{x}_1 \vee \bar{x}_2\bar{x}_3$
NAND	7	$\bar{x}_1(\bar{x}_2 \vee \bar{x}_3)$



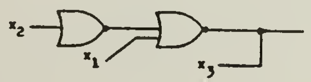
No. 7

NOR	77	$\bar{x}_1 \vee \bar{x}_2$
NAND	3	$\bar{x}_1\bar{x}_2$



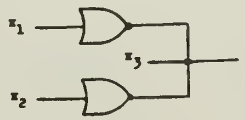
No. 8

NOR	256	$\bar{x}_1x_2 \vee x_3$
NAND	212	$(\bar{x}_1 \vee x_2)x_3$



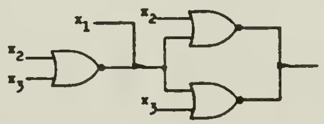
No. 9

NOR	277	$\bar{x}_1 \vee \bar{x}_2 \vee x_3$
NAND	2	$\bar{x}_1\bar{x}_2x_3$



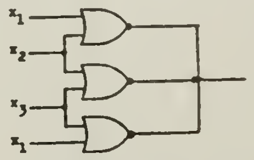
No. 10

NOR	6	$\bar{x}_1(\bar{x}_2x_3 \vee x_2\bar{x}_3)$
NAND	237	$\bar{x}_1 \vee x_2x_3 \vee \bar{x}_2\bar{x}_3$



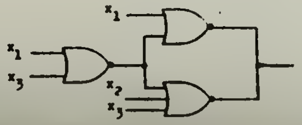
No. 11

NOR	27	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3$
NAND	27	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3$



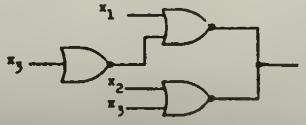
No. 12

NOR	32	$\bar{x}_1x_3 \vee x_1\bar{x}_2\bar{x}_3$
NAND	247	$\bar{x}_1\bar{x}_2 \vee x_1x_3 \vee \bar{x}_1\bar{x}_3$



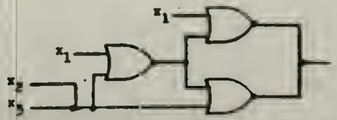
No. 13

NOR	33	$\bar{x}_1x_3 \vee \bar{x}_2\bar{x}_3$
NAND	47	$\bar{x}_1\bar{x}_3 \vee \bar{x}_2x_3$



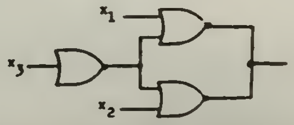
No. 14

NOR	36	$\bar{x}_1(x_2 \vee x_3) \vee x_1\bar{x}_2\bar{x}_3$
NAND	207	$\bar{x}_1(\bar{x}_2 \vee \bar{x}_3) \vee x_1x_2x_3$



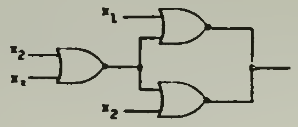
No. 15

NOR	52	$(\bar{x}_1 \vee \bar{x}_2)x_3$
NAND	253	$\bar{x}_1\bar{x}_2 \vee x_3$



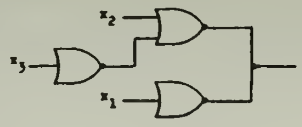
No. 16

	FCT	EXPRESSION
NOR	56	$\bar{x}_1x_2 \vee \bar{x}_2x_3$
NAND	213	$\bar{x}_1\bar{x}_2 \vee x_2x_3$



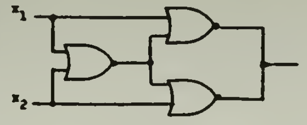
Network no. 17

	FCT	EXPRESSION
NOR	57	$\bar{x}_1 \vee \bar{x}_2x_3$
NAND	13	$\bar{x}_1(\bar{x}_2 \vee x_3)$



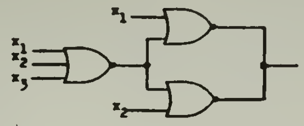
Network no. 18

	FCT	EXPRESSION
NOR	74	$x_1\bar{x}_2 \vee \bar{x}_1x_2$
NAND	303	$x_1x_2 \vee \bar{x}_1\bar{x}_2$



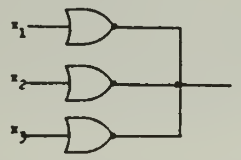
Network no. 19

	FCT	EXPRESSION
NOR	76	$\bar{x}_1x_2 \vee x_1\bar{x}_2 \vee \bar{x}_1x_3$
NAND	203	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2$



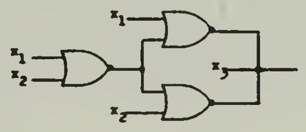
Network no. 20

NOR	177	$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$
NAND	1	$\bar{x}_1\bar{x}_2\bar{x}_3$



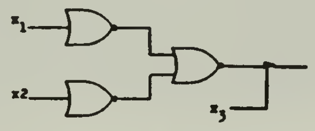
No. 21

NOR	276	$\bar{x}_1x_2 \vee x_1\bar{x}_2 \vee x_3$
NAND	202	$(x_1x_2 \vee \bar{x}_1\bar{x}_2)x_3$



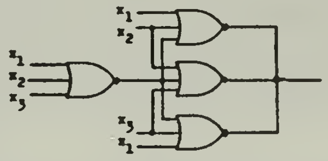
No. 22

NOR	352	$x_1x_2 \vee x_3$
NAND	250	$(x_1 \vee x_2)x_3$



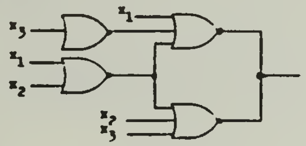
No. 23

NOR	26	$\bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3$
NAND	227	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee x_1x_2x_3$



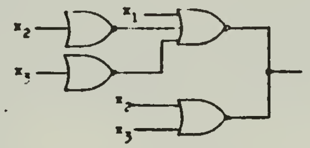
No. 24

NOR	30	$x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2x_3$
NAND	347	$x_1x_2 \vee \bar{x}_2x_3 \vee \bar{x}_1\bar{x}_3$



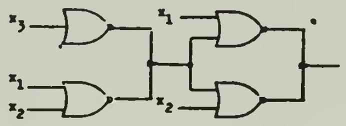
No. 25

NOR	31	$\bar{x}_1x_2x_3 \vee \bar{x}_2\bar{x}_3$
NAND	147	$\bar{x}_1\bar{x}_3 \vee x_2\bar{x}_3 \vee \bar{x}_2x_3$



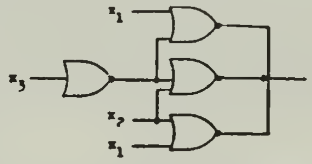
No. 26

NOR	.50	$\bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3$
NAND	353	$\bar{x}_1\bar{x}_2 \vee x_1x_2 \vee x_3$



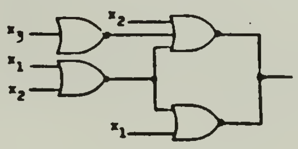
No. 27

NOR	53	$\bar{x}_1\bar{x}_2 \vee (\bar{x}_1 \vee \bar{x}_2)x_3$
NAND	53	$\bar{x}_1\bar{x}_2 \vee (\bar{x}_1 \vee \bar{x}_2)x_3$



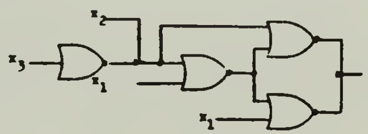
No. 28

NOR	54	$\bar{x}_1x_2 \vee x_1\bar{x}_2x_3$
NAND	313	$x_1x_2 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3$



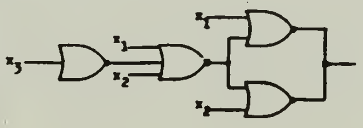
No. 29

NOR	55	$\bar{x}_1(x_2 \vee \bar{x}_3) \vee x_1\bar{x}_2x_3$
NAND	213	$\bar{x}_1x_3 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_2\bar{x}_3$



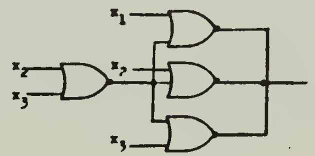
No. 30

NOR	75	$\bar{x}_1x_2 \vee x_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3$
NAND	103	$\bar{x}_1\bar{x}_2 \vee x_1x_2\bar{x}_3$

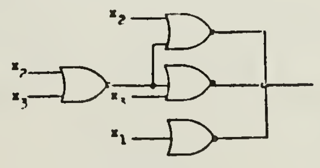
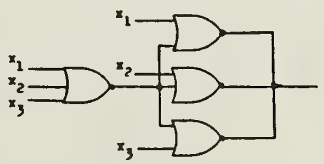
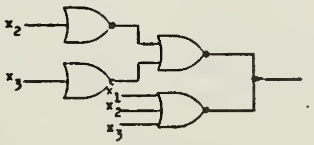
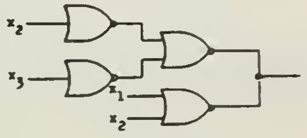
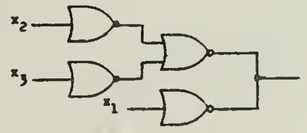
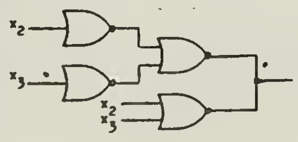
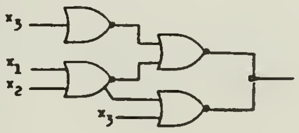
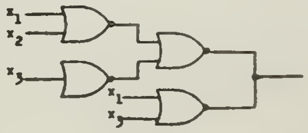
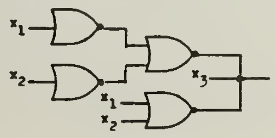
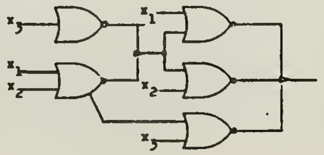
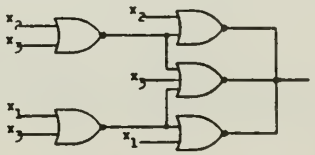
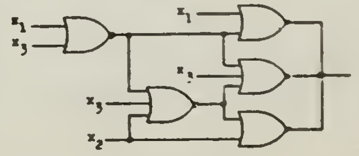
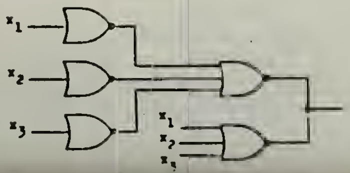
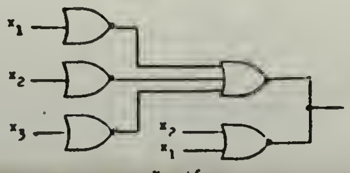
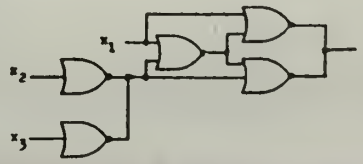
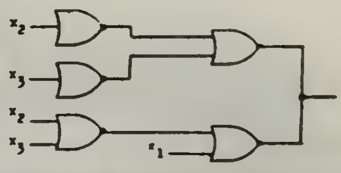


No. 31

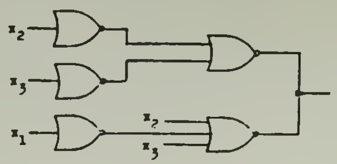
NOR	156	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee \bar{x}_2x_3 \vee x_2\bar{x}_3$
NAND	211	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_2x_3$



No. 32

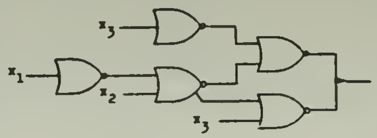
	FCT	EXPRESSION		FCT	EXPRESSION		FCT	EXPRESSION		FCT	EXPRESSION
NOR	157	$\bar{x}_1 \vee \bar{x}_2x_3 \vee x_2\bar{x}_3$	NOR	176	$x_1\bar{x}_2 \vee x_2\bar{x}_3 \vee \bar{x}_1x_3$	NOR	211	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_2x_3$	NOR	213	$\bar{x}_1\bar{x}_2 \vee x_2x_3$
NAND	11	$\bar{x}_1(\bar{x}_2\bar{x}_3 \vee x_2x_3)$	NAND	201	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$	NAND	156	$\bar{x}_1(x_2 \vee x_3) \vee \bar{x}_2x_3 \vee x_2\bar{x}_3$	NAND	56	$\bar{x}_1x_2 \vee \bar{x}_2x_3$
 Network no. 33			 Network no. 34			 Network no. 35			 Network no. 36		
NOR	217	$\bar{x}_1 \vee x_2x_3$	NOR	231	$x_2x_3 \vee \bar{x}_2\bar{x}_3$	NOR	251	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1x_3 \vee x_2x_3$	NOR	255	$\bar{x}_1\bar{x}_3 \vee x_2x_3 \vee x_1x_3$
NAND	16	$\bar{x}_1(x_2 \vee x_3)$	NAND	146	$\bar{x}_2x_3 \vee x_2\bar{x}_3$	NAND	152	$\bar{x}_1x_3 \bar{\vee} x_2x_3 \vee x_1x_2\bar{x}_3$	NAND	·212	$x_2x_3 \vee \bar{x}_1x_3$
 No. 37			 No. 38			 No. 39			 No. 40		
NOR	353	$\bar{x}_1\bar{x}_2 \vee x_1x_2 \vee x_3$	NOR	51	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3$	NOR	152	$\bar{x}_1x_3 \vee x_2\bar{x}_3 \vee x_1x_2\bar{x}_3$	NOR	153	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3 \vee \bar{x}_2x_3 \vee x_1x_2\bar{x}_3$
NAND	50	$(\bar{x}_1x_2 \vee x_1\bar{x}_2)x_3$	NAND	153	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3 \vee \bar{x}_2x_3 \vee x_1x_2\bar{x}_3$	NAND	251	$x_1x_3 \vee x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$	NAND	51	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3$
 No. 41			 No. 42			 No. 43			 No. 44		
NOR	201	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$	NOR	203	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2$	NOR	207	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee x_1x_2x_3$	NOR	216	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3$
NAND	176	$x_1\bar{x}_2 \vee x_2\bar{x}_3 \vee \bar{x}_1x_3$	NAND	76	$\bar{x}_1x_2 \vee x_1\bar{x}_2 \vee \bar{x}_1x_3$	NAND	36	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_1\bar{x}_2\bar{x}_3$	NAND	216	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3$
 No. 45			 No. 46			 No. 47			 No. 48		

	FCT	EXPRESSION
NOR	230	$x_1\bar{x}_2\bar{x}_3 \vee x_2x_3$
NAND	346	$x_1x_2 \vee x_2\bar{x}_3 \vee \bar{x}_2x_3$



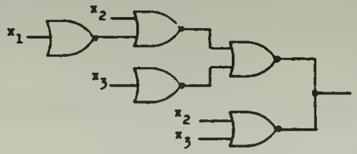
Network no. 49

	FCT	EXPRESSION
NOR	232	$(\bar{x}_1 \vee x_2)x_3 \vee \bar{x}_1x_2\bar{x}_3$
NAND	246	$x_1x_3 \vee \bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3$



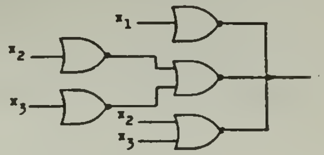
Network no. 50

	FCT	EXPRESSION
NOR	233	$\bar{x}_1\bar{x}_2 \vee \bar{x}_2\bar{x}_3 \vee x_2x_3$
NAND	46	$\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3$



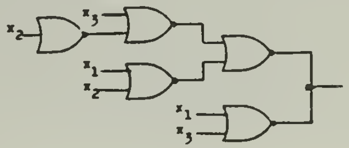
Network oo. 51

	FCT	EXPRESSION
NOR	237	$\bar{x}_1 \vee \bar{x}_2\bar{x}_3 \vee x_2x_3$
NAND	6	$\bar{x}_1(\bar{x}_2x_3 \vee x_2\bar{x}_3)$



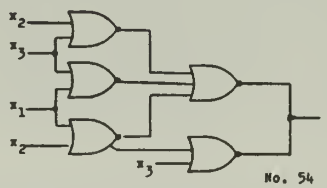
Network oo. 52

	FCT	EXPRESSION
NOR	275	$x_1(\bar{x}_2 \vee x_3) \vee \bar{x}_1(x_2 \vee \bar{x}_3)$
NAND	202	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2x_3$



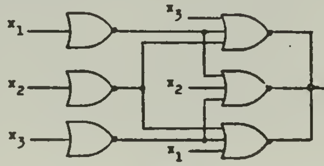
No. 53

	FCT	EXPRESSION
NOR	351	$x_1x_2 \vee x_1x_3 \vee x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$
NAND	150	$\bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3$



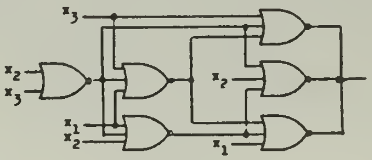
No. 54

	FCT	EXPRESSION
NOR	150	$\bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3$
NAND	351	$x_1x_2 \vee x_1x_3 \vee x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$



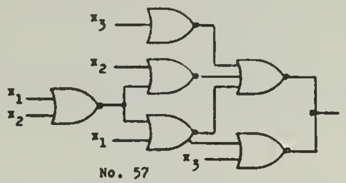
No. 55

	FCT	EXPRESSION
NOR	151	$\bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$
NAND	151	$\bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$



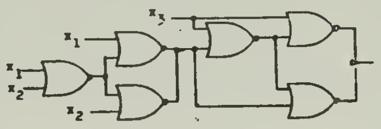
No. 56

	FCT	EXPRESSION
NOR	206	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3$
NAND	236	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3 \vee x_1\bar{x}_2\bar{x}_3$



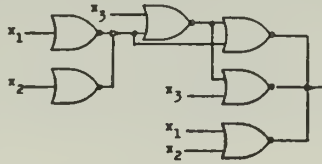
No. 57

	FCT	EXPRESSION
NOR	207	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee x_1x_2x_3$
NAND	36	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_1\bar{x}_2\bar{x}_3$



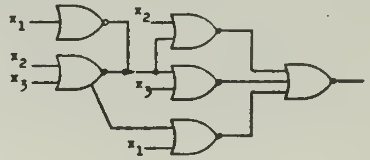
No. 58

	FCT	EXPRESSION
NOR	227	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee x_1x_2x_3$
NAND	26	$\bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3$



No. 59

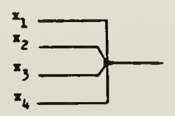
	FCT	EXPRESSION
NOR	236	$\bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3 \vee x_1\bar{x}_2\bar{x}_3$
NAND	206	$x_1x_2x_3 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3$



No. 60

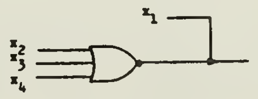
A-2 Functions of four variables

	FCT	EXPRESSION
NOR	7FFF	$x_1 \vee x_2 \vee x_3 \vee x_4$
NAND	1	$x_1x_2x_3x_4$



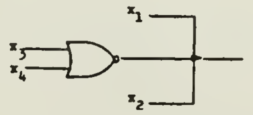
Network no. 1

	FCT	EXPRESSION
NOR	80FF	$x_1 \vee \bar{x}_2\bar{x}_3\bar{x}_4$
NAND	FE	$x_1(\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)$



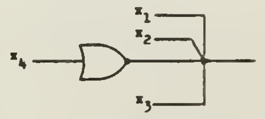
Network no. 2

	FCT	EXPRESSION
NOR	8FFF	$x_1 \vee x_2 \vee \bar{x}_3\bar{x}_4$
NAND	E	$x_1x_2(\bar{x}_3 \vee \bar{x}_4)$



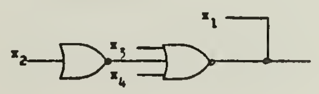
Network no. 3

	FCT	EXPRESSION
NOR	BFFF	$x_1 \vee x_2 \vee x_3 \vee \bar{x}_4$
NAND	2	$x_1x_2x_3\bar{x}_4$



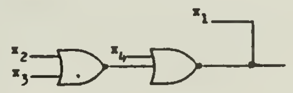
Network no. 4

	FCT	EXPRESSION
NOR	8FF	$x_1 \vee x_2\bar{x}_3\bar{x}_4$
NAND	EF	$x_1(x_2 \vee \bar{x}_3 \vee \bar{x}_4)$



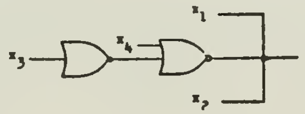
No. 5

	FCT	EXPRESSION
NOR	2AFF	$x_1 \vee (x_2 \vee x_3)\bar{x}_4$
NAND	AB	$x_1x_2x_3 \vee x_1\bar{x}_4$



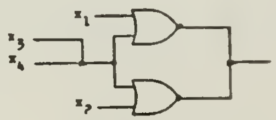
No. 6

	FCT	EXPRESSION
NOR	2FFF	$x_1 \vee x_2 \vee x_3\bar{x}_4$
NAND	B	$x_1x_2(x_3 \vee \bar{x}_4)$



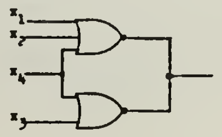
No. 7

	FCT	EXPRESSION
NOR	8880	$(\bar{x}_1 \vee \bar{x}_2)\bar{x}_3\bar{x}_4$
NAND	FEEE	$\bar{x}_1\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$



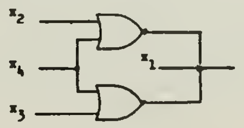
No. 8

	FCT	EXPRESSION
NOR	A888	$\bar{x}_1\bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4$
NAND	EEEA	$\bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$



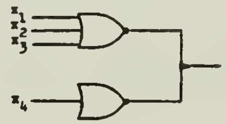
No. 9

	FCT	EXPRESSION
NOR	A8FF	$x_1 \vee (\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$
NAND	EA	$x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_4$



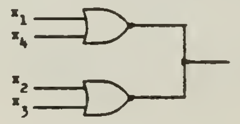
No. 10

	FCT	EXPRESSION
NOR	EAAA	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_4$
NAND	AAA8	$(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



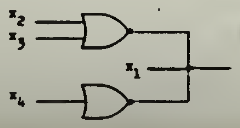
No. 11

	FCT	EXPRESSION
NOR	EAC0	$\bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3$
NAND	FCA8	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4$



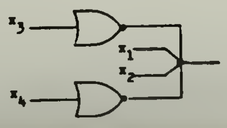
No. 12

	FCT	EXPRESSION
NOR	EAFF	$x_1 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$
NAND	A8	$x_1(\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



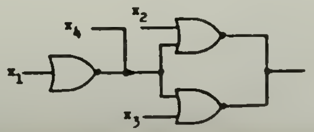
No. 13

	FCT	EXPRESSION
NOR	EFFF	$x_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4$
NAND	8	$x_1x_2\bar{x}_3\bar{x}_4$



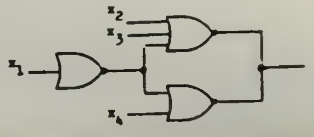
No. 14

	FCT	EXPRESSION
NOR	A8	$x_1(\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$
NAND	EAFF	$x_1 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$



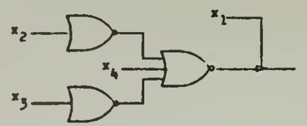
No. 15

	FCT	EXPRESSION
NOR	EA	$x_1(\bar{x}_2\bar{x}_3 \vee \bar{x}_4)$
NAND	A8FF	$x_1 \vee (\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



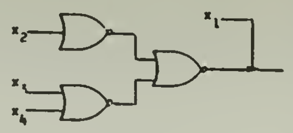
No. 16

	FCT	EXPRESSION
NOR	2FF	$x_1 \vee x_2x_3\bar{x}_4$
NAND	BF	$x_1(x_2 \vee x_3 \vee \bar{x}_4)$



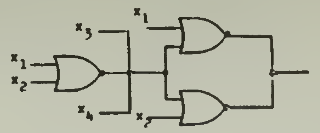
Network no. 17

	FCT	EXPRESSION
NOR	7FF	$x_1 \vee x_2(x_3 \vee x_4)$
NAND	1F	$x_1(x_2 \vee x_3x_4)$



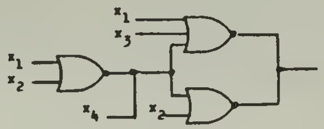
Network no. 18

	FCT	EXPRESSION
NOR	880	$(\bar{x}_1x_2 \vee x_1\bar{x}_2)\bar{x}_3\bar{x}_4$
NAND	FEEF	$x_1x_2 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$



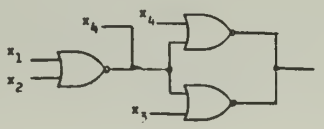
Network no. 19

	FCT	EXPRESSION
NOR	8A0	$x_1\bar{x}_2\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4$
NAND	FAEF	$x_1x_2 \vee \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee \bar{x}_4$



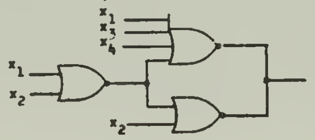
Network no. 20

	FCT	EXPRESSION
NOR	8A8	$x_1\bar{x}_2\bar{x}_4 \vee x_2\bar{x}_3\bar{x}_4$
NAND	EAEF	$x_1x_2 \vee x_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$



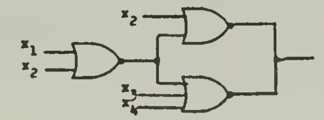
No. 21

	FCT	EXPRESSION
NOR	8F0	$x_1\bar{x}_2 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4$
NAND	F0EF	$x_1(x_2 \vee \bar{x}_3 \vee \bar{x}_4) \vee \bar{x}_1\bar{x}_2$



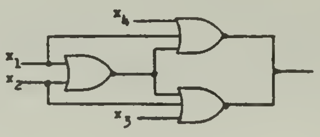
No. 22

	FCT	EXPRESSION
NOR	8F8	$x_1\bar{x}_2 \vee x_2\bar{x}_3\bar{x}_4$
NAND	E0EF	$x_1(x_2 \vee \bar{x}_3) \vee \bar{x}_2(\bar{x}_3 \vee \bar{x}_4)$



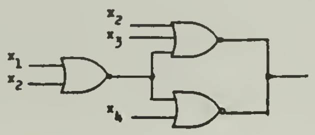
No. 23

	FCT	EXPRESSION
NOR	AC0	$\bar{x}_1x_2\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3$
NAND	FCAF	$x_1x_2 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4$



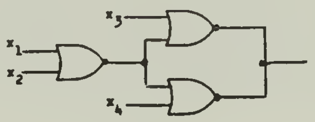
No. 24

	FCT	EXPRESSION
NOR	AEA	$\bar{x}_1x_2\bar{x}_4 \vee x_1\bar{x}_2(\bar{x}_3 \vee \bar{x}_4)$
NAND	A8AF	$x_1x_2 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4$



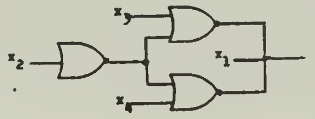
No. 25

	FCT	EXPRESSION
NOR	EEE	$(x_1 \vee x_2)(\bar{x}_3 \vee \bar{x}_4)$
NAND	888F	$x_1x_2 \vee \bar{x}_3\bar{x}_4$



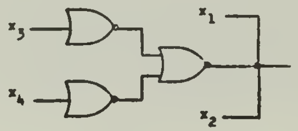
No. 26

	FCT	EXPRESSION
NOR	EFF	$x_1 \vee x_2(\bar{x}_3 \vee \bar{x}_4)$
NAND	8F	$x_1(x_2 \vee \bar{x}_3\bar{x}_4)$



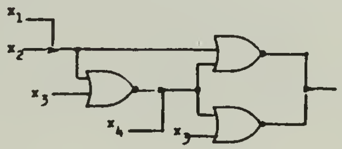
No. 27

	FCT	EXPRESSION
NOR	1FFF	$x_1 \vee x_2 \vee x_3x_4$
NAND	7	$x_1x_2(x_3 \vee x_4)$



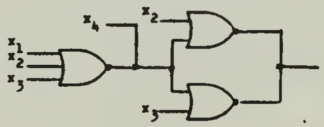
No. 28

	FCT	EXPRESSION
NOR	2888	$(x_1 \vee x_2)\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4$
NAND	EEEB	$x_1x_2x_3 \vee (\bar{x}_1 \vee \bar{x}_2)\bar{x}_3 \vee \bar{x}_4$



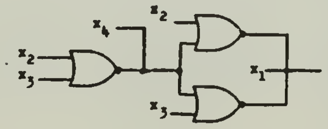
No. 29

	FCT	EXPRESSION
NOR	28A8	$(x_1 \vee x_2)\bar{x}_3\bar{x}_4 \vee \bar{x}_2x_3\bar{x}_4$
NAND	EAEB	$x_1x_2x_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$



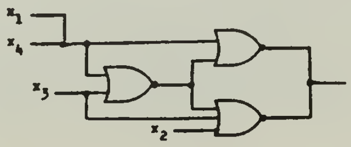
No. 30

	FCT	EXPRESSION
NOR	28FF	$x_1 \vee (\bar{x}_2x_3 \vee x_2\bar{x}_3)\bar{x}_4$
NAND	EB	$x_1(x_2x_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4)$



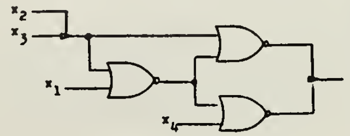
No. 31

	FCT	EXPRESSION
NOR	2AC0	$x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_4 \vee \bar{x}_1x_3\bar{x}_4$
NAND	FCAB	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4 \vee x_1x_2x_3$



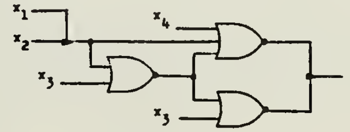
No. 32

	FCT	EXPRESSION
NOR	2AEA	$x_1\bar{x}_2\bar{x}_3 \vee (x_2 \vee x_3)\bar{x}_4$
NAND	A8AB	$x_1x_2x_3 \vee (\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



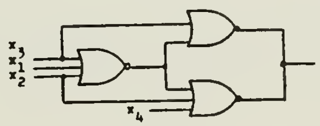
Network no. 33

	FCT	EXPRESSION
NOR	2CCC	$(x_1 \vee x_2)\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4$
NAND	CCCB	$(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4)\bar{x}_3 \vee x_1x_2x_3$



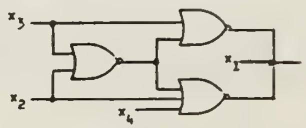
Network no. 34

	FCT	EXPRESSION
NOR	2CEC	$(x_1 \vee x_2)\bar{x}_3 \vee \bar{x}_2x_3\bar{x}_4$
NAND	C8CB	$(\bar{x}_2 \vee \bar{x}_4)\bar{x}_3 \vee x_1x_2x_3$



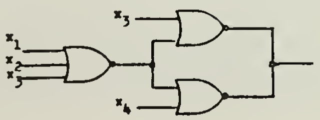
Network no. 35

	FCT	EXPRESSION
NOR	2CFF	$x_1 \vee x_2\bar{x}_3 \vee \bar{x}_2x_3\bar{x}_4$
NAND	CB	$x_1(x_2x_3 \vee \bar{x}_2\bar{x}_3 \vee x_2\bar{x}_4)$



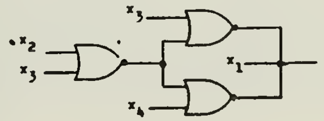
Network no. 36

NOR	2EEE	$(x_1 \vee x_2)\bar{x}_3 \vee (x_2 \vee x_3)\bar{x}_4$
NAND	888B	$x_1x_2x_3 \vee \bar{x}_3\bar{x}_4$



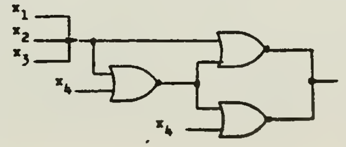
No. 37

NOR	2EFF	$x_1 \vee x_2\bar{x}_3 \vee x_3\bar{x}_4$
NAND	8B	$x_1(x_2x_3 \vee \bar{x}_3\bar{x}_4)$



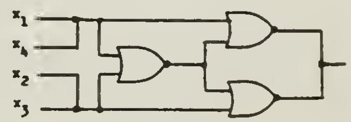
No. 38

NOR	6AAA	$(x_1 \vee x_2 \vee x_3)\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4$
NAND	AAA9	$(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)\bar{x}_4 \vee x_1x_2x_3x_4$



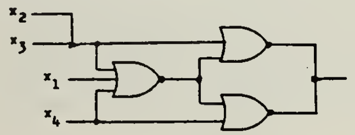
No. 39

NOR	6AC0	$x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_4 \vee \bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4$
NAND	FCA9	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3x_4 \vee x_1x_2x_3x_4$



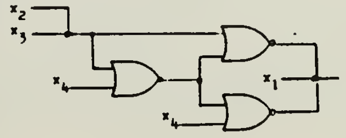
No. 40

NOR	6AEA	$(x_1 \vee x_2 \vee x_3)\bar{x}_4 \vee \bar{x}_2\bar{x}_3x_4$
NAND	A8A9	$(\bar{x}_2 \vee \bar{x}_3)\bar{x}_4 \vee x_1x_2x_3x_4$



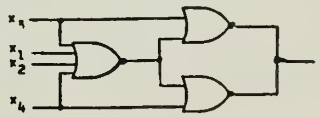
No. 41

NOR	6AFF	$x_1 \vee (x_2 \vee x_3)\bar{x}_4 \vee \bar{x}_2\bar{x}_3x_4$
NANE	A9	$x_1(\bar{x}_2 \vee \bar{x}_3)\bar{x}_4 \vee x_1x_2x_3x_4$



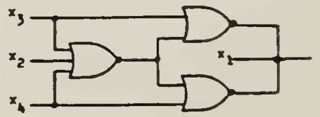
No. 42

NOR	6EEE	$x_1\bar{x}_3 \vee x_2\bar{x}_4 \vee \bar{x}_3x_4 \vee x_3\bar{x}_4$
NAND	8889	$\bar{x}_3\bar{x}_4 \vee x_1x_2x_3x_4$



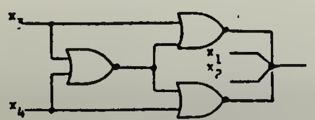
No. 43

NOR	6EFF	$x_1 \vee (x_2 \vee x_3)\bar{x}_4 \vee \bar{x}_3x_4$
NAND	89	$x_1(\bar{x}_3\bar{x}_4 \vee x_2x_3x_4)$



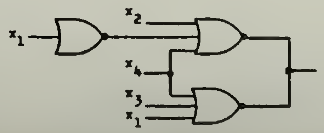
No. 44

NOR	6FFF	$x_1 \vee x_2 \vee \bar{x}_3x_4 \vee x_3\bar{x}_4$
NAND	9	$x_1x_2(x_3x_4 \vee \bar{x}_3\bar{x}_4)$



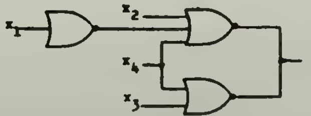
No. 45

NOR	88A0	$(\bar{x}_1 \vee \bar{x}_2)\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_4$
NAND	FAEE	$\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee \bar{x}_4$



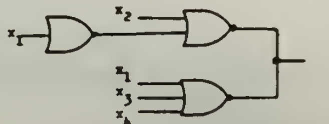
No. 46

NOR	88A8	$\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_4$
NAND	EAEE	$x_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$



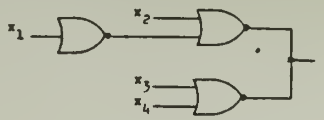
No. 47

NOR	88F0	$x_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3\bar{x}_4$
NAND	F0EE	$\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee x_1\bar{x}_4$



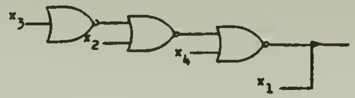
No. 48

	FCT	EXPRESSION
NOR	88F8	$x_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4$
NAND	EOEE	$(x_1 \vee \bar{x}_2)(\bar{x}_3 \vee \bar{x}_4)$



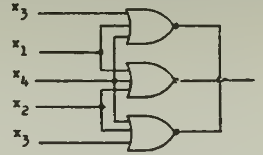
Network no. 49

	FCT	EXPRESSION
NOR	8AFF	$x_1 \vee (x_2 \vee \bar{x}_3)\bar{x}_4$
NAND	AE	$x_1\bar{x}_4 \vee x_1x_2\bar{x}_3$



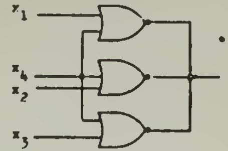
Network no. 50

	FCT	EXPRESSION
NOR	A880	$\bar{x}_1\bar{x}_2\bar{x}_4 \vee \bar{x}_1\bar{x}_3\bar{x}_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4$
NAND	FEEA	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$



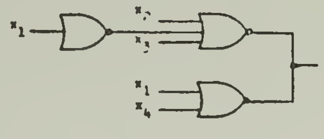
Network no. 51

	FCT	EXPRESSION
NOR	AAA8	$\bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4$
NAND	EAAA	$\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_4$



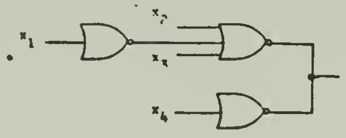
Network no. 52

	FCT	EXPRESSION
NOR	AACO	$x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_4$
NAND	FCAA	$\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee x_1\bar{x}_4$



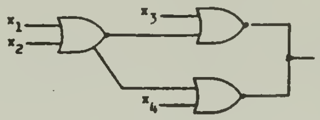
No. 53

	FCT	EXPRESSION
NOR	AAEA	$x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_4$
NAND	A8AA	$(x_1 \vee \bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



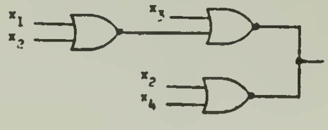
No. 54

	FCT	EXPRESSION
NOR	ACCC	$(x_1 \vee x_2)\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_4$
NAND	CCCA	$\bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee x_1x_2\bar{x}_4$



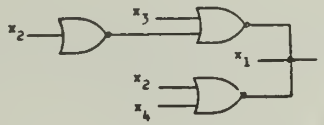
No. 55

	FCT	EXPRESSION
NOR	ACEC	$(x_1 \vee x_2)\bar{x}_3 \vee \bar{x}_2\bar{x}_4$
NAND	C8CA	$\bar{x}_2\bar{x}_3 \vee \bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_4$



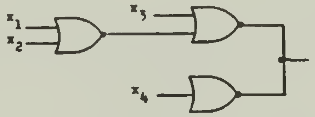
No. 56

	FCT	EXPRESSION
NOR	ACFF	$x_1 \vee x_2\bar{x}_3 \vee \bar{x}_2\bar{x}_4$
NAND	CA	$x_1(\bar{x}_2\bar{x}_3 \vee x_2\bar{x}_4)$



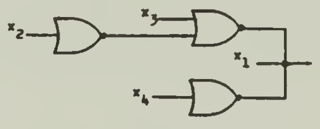
No. 57

	FCT	EXPRESSION
NOR	AEEE	$(x_1 \vee x_2)x_3 \vee \bar{x}_4$
NAND	888A	$(x_1x_2 \vee x_3)\bar{x}_4$



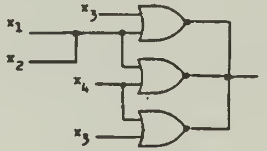
No. 58

	FCT	EXPRESSION
NOR	AEFF	$x_1 \vee x_2\bar{x}_3 \vee \bar{x}_4$
NAND	8A	$x_1(x_2 \vee \bar{x}_3)\bar{x}_4$



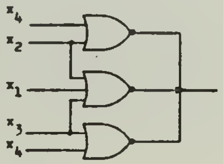
No. 59

	FCT	EXPRESSION
NOR	E888	$\bar{x}_1\bar{x}_2(\bar{x}_3 \vee \bar{x}_4) \vee \bar{x}_3\bar{x}_4$
NAND	EEE8	$\bar{x}_1(\bar{x}_3 \vee \bar{x}_4) \vee (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



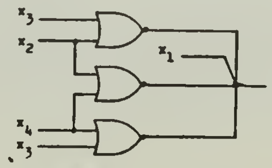
No. 60

	FCT	EXPRESSION
NOR	E8A8	$\bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3$
NAND	EAE8	$(\bar{x}_1 \vee \bar{x}_2)\bar{x}_4 \vee (\bar{x}_2 \vee \bar{x}_4)\bar{x}_3$



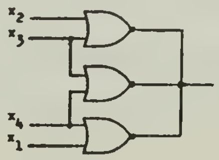
No. 61

	FCT	EXPRESSION
NOR	E8FF	$x_1 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4$
NAND	E8	$x_1(\bar{x}_2\bar{x}_3 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4)$



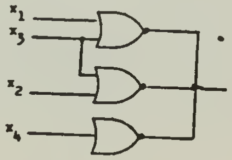
No. 62

	FCT	EXPRESSION
NOR	EAC8	$\bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_3\bar{x}_4$
NAND	ECA8	$\bar{x}_1\bar{x}_3 \vee (\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$

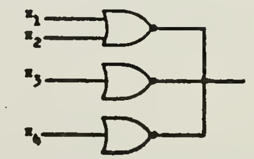
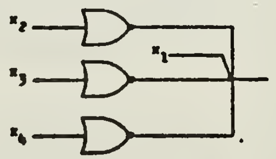


No. 63

	FCT	EXPRESSION
NOR	EEEA	$\bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_4$
NAND	A888	$(\bar{x}_1\bar{x}_2 \vee \bar{x}_3)\bar{x}_4$



No. 64

	FCT	EXPRESSION		FCT	EXPRESSION						
NOR	FEEE	$\bar{x}_1\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$	NOR	FEFF	$x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$						
NAND	8880	$(\bar{x}_1 \vee \bar{x}_2)\bar{x}_3\bar{x}_4$	NAND	80	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$						
 Network no. 65			 Network no. 66								

APPENDIX B

Statistics on Optimal Networks

Statistics on optimal networks for each of all representative functions of three or less variables except trivial functions (i.e., 0, 1 and X_i) are shown in B-1. Statistics on optimal networks for each of all representative functions of four variables which can be implemented with four or less NOR gates in the original network derived by Phase 1 and for each of all representative functions of four variables whose numbers of NOR gates are reduced from 5 to 3 by the algorithm are shown in B-2.

Original networks are those obtained by Phase 1 of the algorithm and optimal networks are those obtained by Phases 2 and 3. Optimal networks shown with "-" sign in the optimal network column are identical to original networks. (These networks do not have network numbers in tables).

Because we do not have gates which have only OR outputs, the numbers of NOR outputs in the optimal network column also express the total numbers of gates in the networks.

B-1 Functions of three or less variables

Network number	FCT	Original networks derived by phase 1: No. of NOR gates	Original networks derived by phase 1: No. of connections	Optimal networks derived by entire algorithm: No. of NOR outputs	Optimal networks derived by entire algorithm: No. of OR outputs	Optimal networks derived by entire algorithm: No. of Wired-ORs	Optimal networks derived by entire algorithm: No. of connections
	1	1	3		-		
	2	2	4		-		
	3	1	2		-		
11	6	5	9	3	0	2	8
	7	4	6		-		
	10	3	5		-		
	11	4	9		-		
	12	2	3		-		
	13	3	5		-		
	16	2	4		-		
	17	1	1		-		
24	26	6	14	4	0	1	14
12	27	5	10	3	0	1	8
25	30	6	11	4	0	1	10
26	31	6	10	4	0	1	8
13	32	5	9	3	0	1	8
14	33	5	8	3	0	1	6
15	36	5	10	3	0	2	8
7	37	4	6	2	0	1	4
27	50	6	10	4	0	2	9
42	51	7	14	5	1	2	12
16	52	5	7	3	0	1	6
28	53	6	10	4	0	1	9
29	54	6	10	4	0	1	9
30	55	6	11	4	0	2	9
17	56	5	8	3	0	1	7
18	57	5	7	3	0	1	5
19	74	5	8	3	0	1	7
31	75	6	10	4	0	1	9
20	76	5	9	3	0	1	8
8	77	4	5	2	0	1	3
55	150	7	15	6	0	1	14
56	151	7	16	6	0	1	19
43	152	6	12	5	0	1	13
44	153	7	15	5	0	1	14
32	156	6	10	4	0	1	10
33	157	6	11	4	0	1	9
34	176	6	11	4	0	1	11
21	177	5	7	3	0	1	5
	200	4	6		-		
45	201	5	12	5	0	1	10
	202	5	10		-		
46	203	5	11	5	0	1	9
57	206	6	15	6	1	1	13
47	207	6	14	5	0	2	10

Network number	FCT	Original networks derived by phase 1		Optimal networks derived by entire algorithm			
		No. of NOR gates	No. of connections	No. of NOR outputs	No. of OR outputs	No. of Wired-ORs	No. of connections
	210	3	4		-		
35	211	4	9	4	0	1	8
	212	4	6		-		
36	213	4	8	4	0	1	7
48	216	5	10	5	0	1	9
37	217	4	7	4	0	1	6
58	226	7	20	6	0	2	14
59	227	7	15	6	0	2	13
49	230	5	10	5	0	1	9
38	231	4	8	4	0	1	7
50	232	5	11	5	1	1	9
51	233	5	10	5	0	1	9
60	236	6	15	6	1	1	13
52	237	5	11	5	0	1	9
	250	3	5		-		
39	251	4	10	4	1	1	8
3	253	3	5	1	0	1	3
	254	4	7		-		
40	255	4	9	4	0	1	8
9	256	4	6	2	0	1	4
4	257	3	4	1	0	1	2
	274	5	9		-		
53	275	5	11	5	0	1	10
22	276	5	10	3	0	1	8
10	277	4	6	2	0	1	4
	350	4	9		-		
54	351	5	15	5	1	1	12
23	352	3	6	3	0	1	5
41	353	4	10	4	0	1	8
1	356	2	3	0	0	1	1
5	357	3	5	1	0	1	3
2	376	2	4	0	0	1	2

B-2 Functions of four variables which require four or less NOR-OR gates in each optimal network

Network number	FCT	Original networks derived by phase 1		Optimal networks derived by entire algorithm			
		No. of NOR gates	No. of connections	No. of NOR outputs	No. of OR outputs	No. of Wired-ORs	No. of connections
	2	4	7		-		
	7	4	7		-		
	8	3	6		-		
	1F	4	8		-		
	2A	3	6		-		
	7F	3	6		-		
	80	2	5		-		
	8A	4	7		-		
	8F	4	7		-		
15	A8	5	8	3	0	2	7
	BF	4	7		-		
16	EA	5	8	3	0	1	7
	1FF	4	9	4	0	1	7
	22A	4	10		-		
	28A	4	8		-		
	2AA	3	7		-		
17	2FF	5	8	3	0	1	6
	35F	4	9		-		
	37F	4	10		-		
	38F	4	8		-		
	777	3	6		-		
	77F	4	11	4	0	2	10
	78F	4	9	4	1	0	8
	7F7	4	8		-		
18	7FF	3	7	3	0	1	6
19	880	5	10	3	0	3	9
	888	2	5		-		
20	8A0	5	10	3	0	2	9
21	8A8	5	9	3	0	2	8
	8AA	4	7		-		
22	8F0	5	10	3	0	1	9
23	8F8	5	9	3	0	1	8
5	8FF	4	7	2	0	1	5
24	AC0	5	10	3	0	1	9
25	AEA	5	9	3	0	1	8
	BBB	4	7		-		
	BFF	4	8	4	0	1	7
26	EEE	5	8	3	0	1	7
27	EFF	5	9	3	0	1	7
	17FF	4	12	4	0	1	10
	1BBB	4	8		-		
	1BFF	4	9	4	0	1	8
28	1FFF	3	8	3	0	1	6

Network number	NCT	Original networks derived by phase 1		Optimal networks derived by entire algorithm			
		No. of NOR gates	No. of connections	No. of NOR outputs	No. of OR outputs	No. of Wired-ORs	No. of connections
29	2888	5	11	3	0	3	9
30	28A8	5	10	3	0	2	9
31	28FF	5	12	3	0	2	9
	2AAA	2	5		-		
32	2AC0	5	11	3	0	2	9
33	2AFA	5	10	3	0	2	8
6	2AFF	4	7	2	0	1	5
34	2CCC	5	11	3	0	2	9
35	2CEC	5	10	3	0	1	9
36	2CFF	5	11	3	0	1	9
37	2EEE	5	9	3	0	1	8
38	2EFF	5	10	3	0	1	8
7	2FFF	4	7	2	0	1	5
39	6AAA	5	11	3	0	2	9
40	6AC0	5	12	3	0	3	9
41	6AFA	5	11	3	0	2	9
42	6AFF	5	12	3	0	2	9
43	6EFE	5	10	3	0	1	9
44	6EFF	5	11	3	0	1	9
45	6FFF	5	11	3	0	1	9
1	7FFF	2	5	0	0	1	3
	8000	1	4		-		
	8008	4	10		-		
	800A	4	10		-		
	800F	4	10	4	0	1	9
	802A	4	11	4	0	1	10
	803F	4	11	4	1	1	9
	807F	4	12	4	1	1	9
	8088	3	6		-		
	808A	4	9		-		
	808F	4	9	4	0	1	8
	80AA	3	6		-		
	80EF	4	10	4	1	1	8
2	80FF	3	6	1	0	1	4
	828A	4	10		-		
	82AA	4	11	4	0	1	10
	838F	4	10	4	0	1	9
	83FF	4	11	4	0	1	10
	8777	4	12	4	1	1	9
	878F	4	11	4	0	2	9
	87F7	4	11	4	1	1	9
	87FF	4	12	4	1	1	9
8	8880	4	7	2	0	2	6
	888A	4	8		-		
	888F	4	8	4	0	1	7
46	88A0	5	9	3	0	1	8
47	88A8	5	8	3	0	1	7

		Original networks derived by phase 1		Optimal networks derived by entire algorithm			
Network number	FCT	No. of NOR gates	No. of connections	No. of NOR outputs	No. of OR outputs	No. of Wired-ORs	No. of connections
	88BF	4	9	4	0	1	8
48	88F0	5	9	3	0	1	7
49	88F8	5	8	3	0	1	6
	8AAA	3	6		-		
50	8AFF	5	8	3	0	1	6
	8BBB	4	9	4	0	1	8
	8BFF	4	10	4	0	1	9
3	8FFF	3	6	1	0	1	4
	9BBB	4	10	4	0	1	9
	9BDF	4	10	4	0	1	9
	9BFF	4	11	4	0	1	9
	9FFF	4	12	4	0	1	9
51	A880	5	11	3	0	1	11
9	A888	4	7	2	0	1	6
	A8AA	4	7		-		
10	A8FF	4	8	2	0	1	6
52	AAA8	5	8	3	0	1	8
	AABF	4	8	4	0	1	7
53	AAC0	5	9	3	0	1	7
54	AAEA	5	8	3	0	1	6
	ABFF	4	9	4	0	1	7
55	ACCC	5	10	3	1	1	7
56	ACEC	5	9	3	0	1	7
57	ACFF	5	9	3	0	1	7
58	AEEE	5	8	3	0	1	6
59	AEFE	5	8	3	0	1	6
4	BFFF	3	6	1	0	1	4
60	E888	5	12	3	0	2	9
61	E8A8	5	11	3	0	1	9
62	E8FF	5	11	3	0	1	9
11	EAAA	4	7	2	0	1	5
12	EAC0	4	7	2	0	1	5
63	EAC8	5	10	3	0	1	8
13	EAFF	4	7	2	0	1	5
64	EEEA	5	9	3	0	1	7
14	EFFF	4	7	2	0	1	5
65	FEFE	5	8	3	0	1	6
66	FEFF	5	8	3	0	1	6

APPENDIX C

Flow Charts

The detailed flow charts for the programs in the program package which consists of one main program and seven subroutines as shown in Table 5.1 are shown here. The step numbers of the algorithm are also shown in the flow charts.

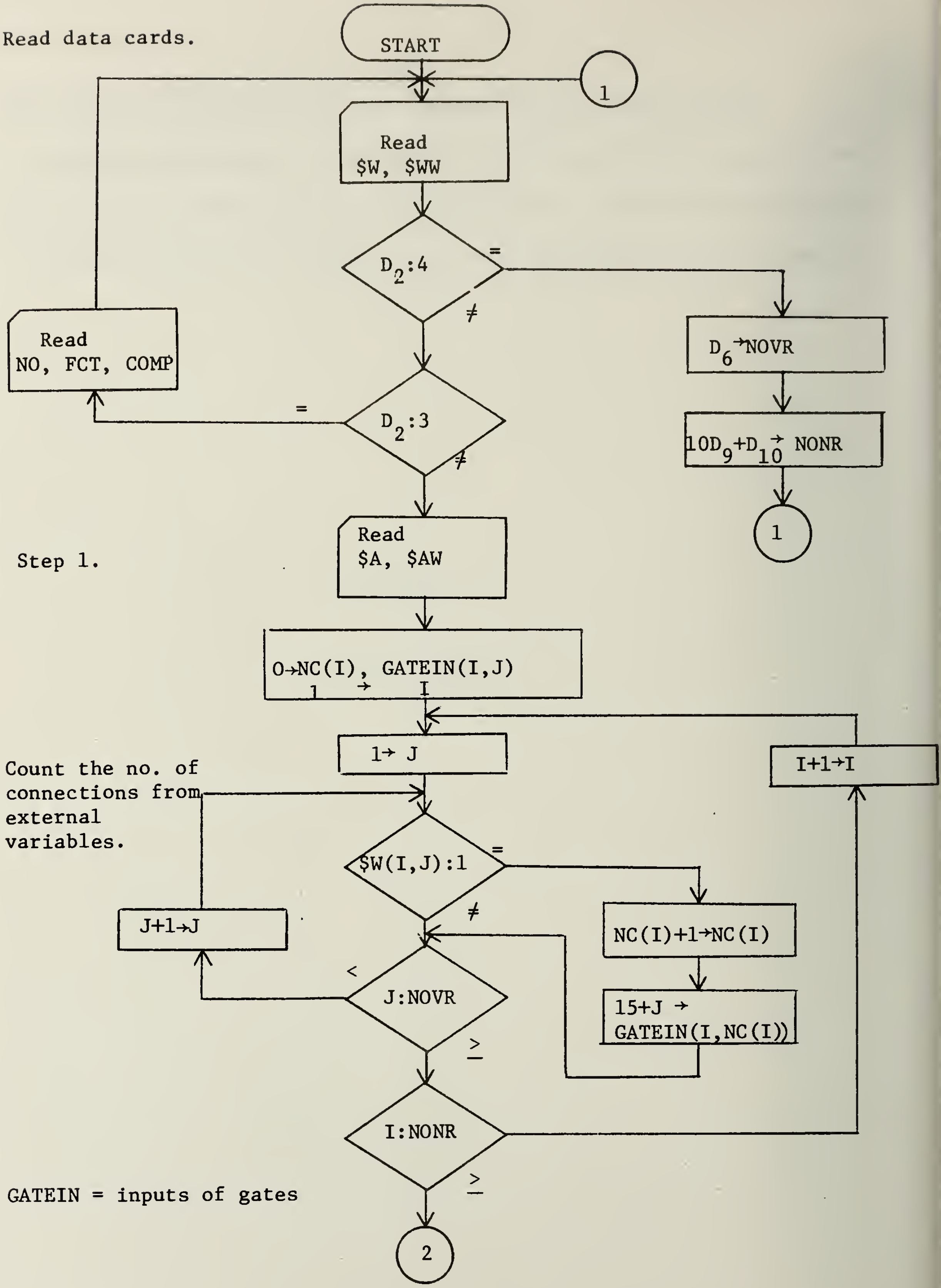
Read data cards.

NOVR = no. of variables.
NONR = no. of NOR gates.

Step 1.

Count the no. of connections from external variables.

GATEIN = inputs of gates

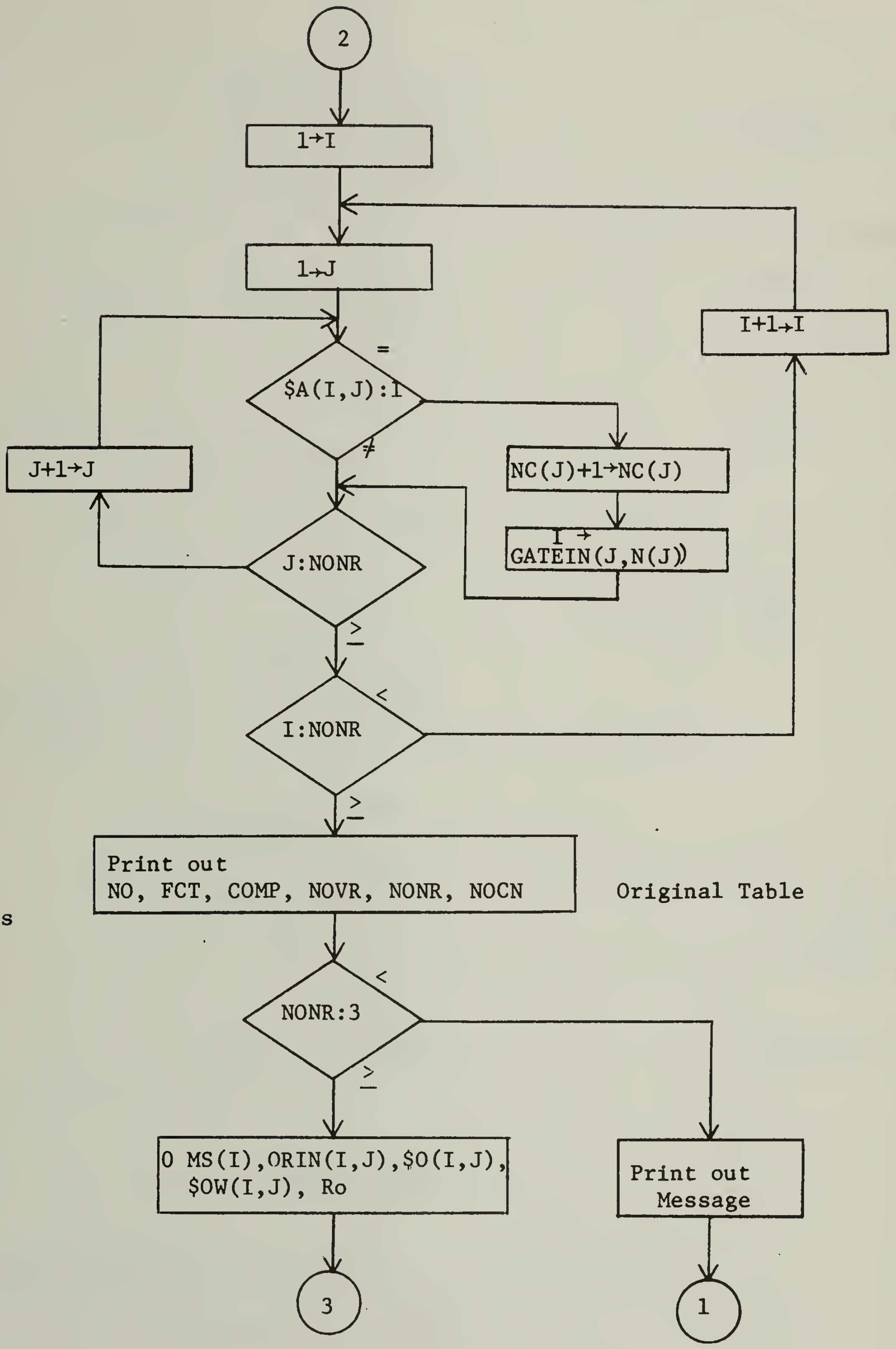


Count the no. of connections from NOR gates.

Print out.

NOCN: no. of connections

Initialization.



Find OR-output.

Step 2.

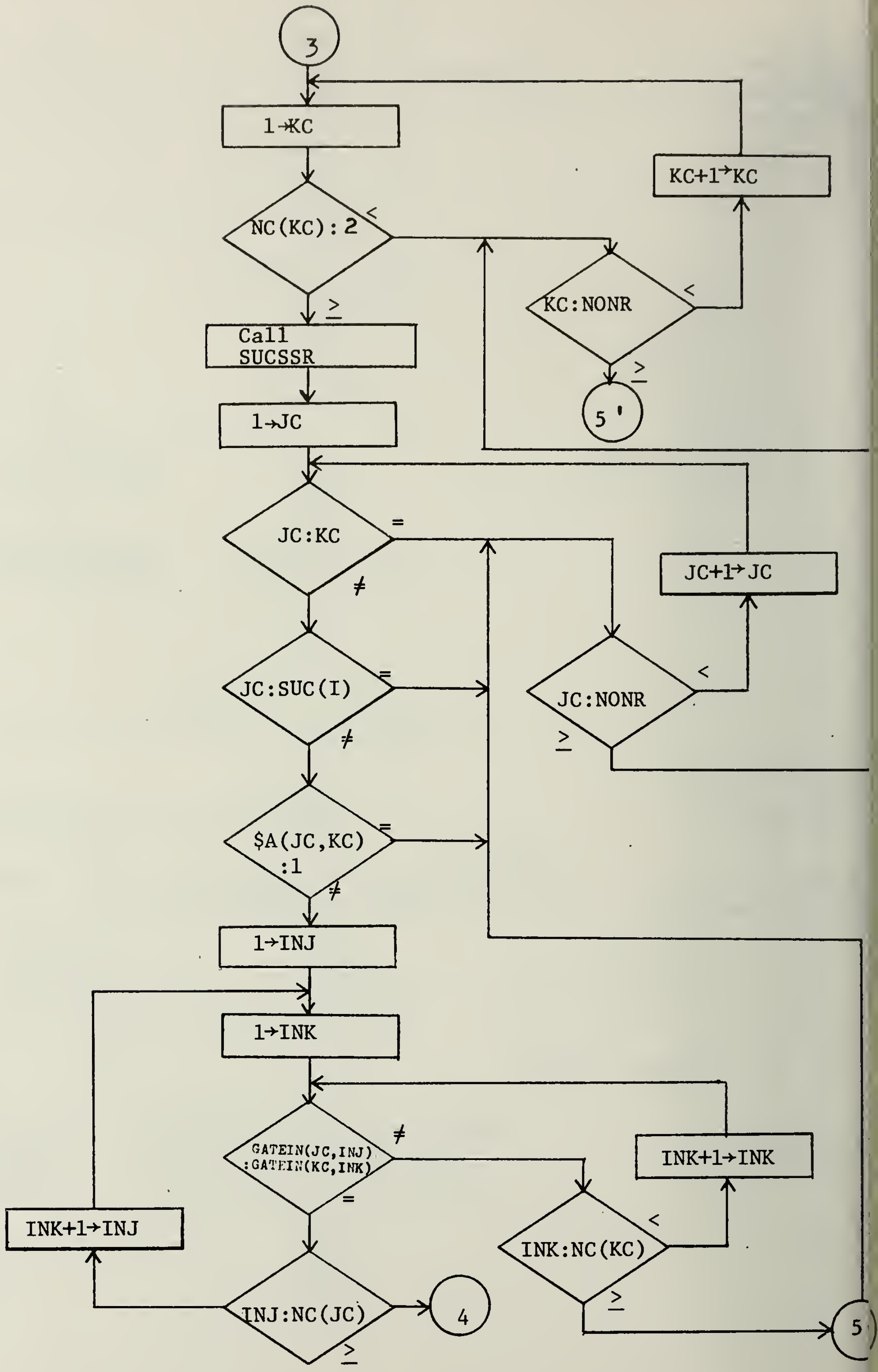
KC: K counter.

Step 3.

JC: J counter.

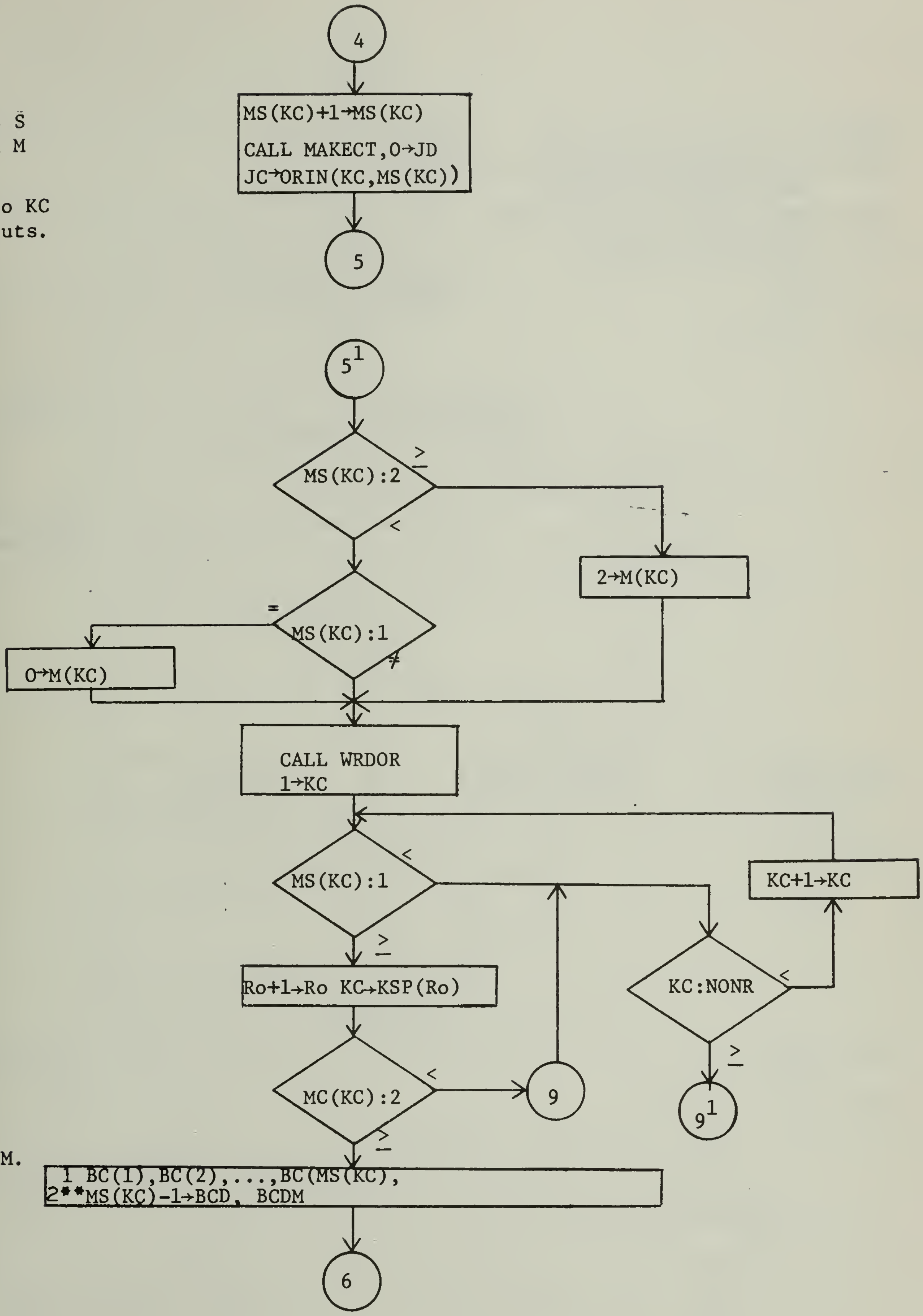
Step 4.

Find Subset.



If MS=1, KC in S
If MS=2, KC in M
Step 5.

ORIN: Inputs to KC
  from OR outputs.
Step 6.



Step 7.

Step 8.

Gate in Group M.
Step 9.

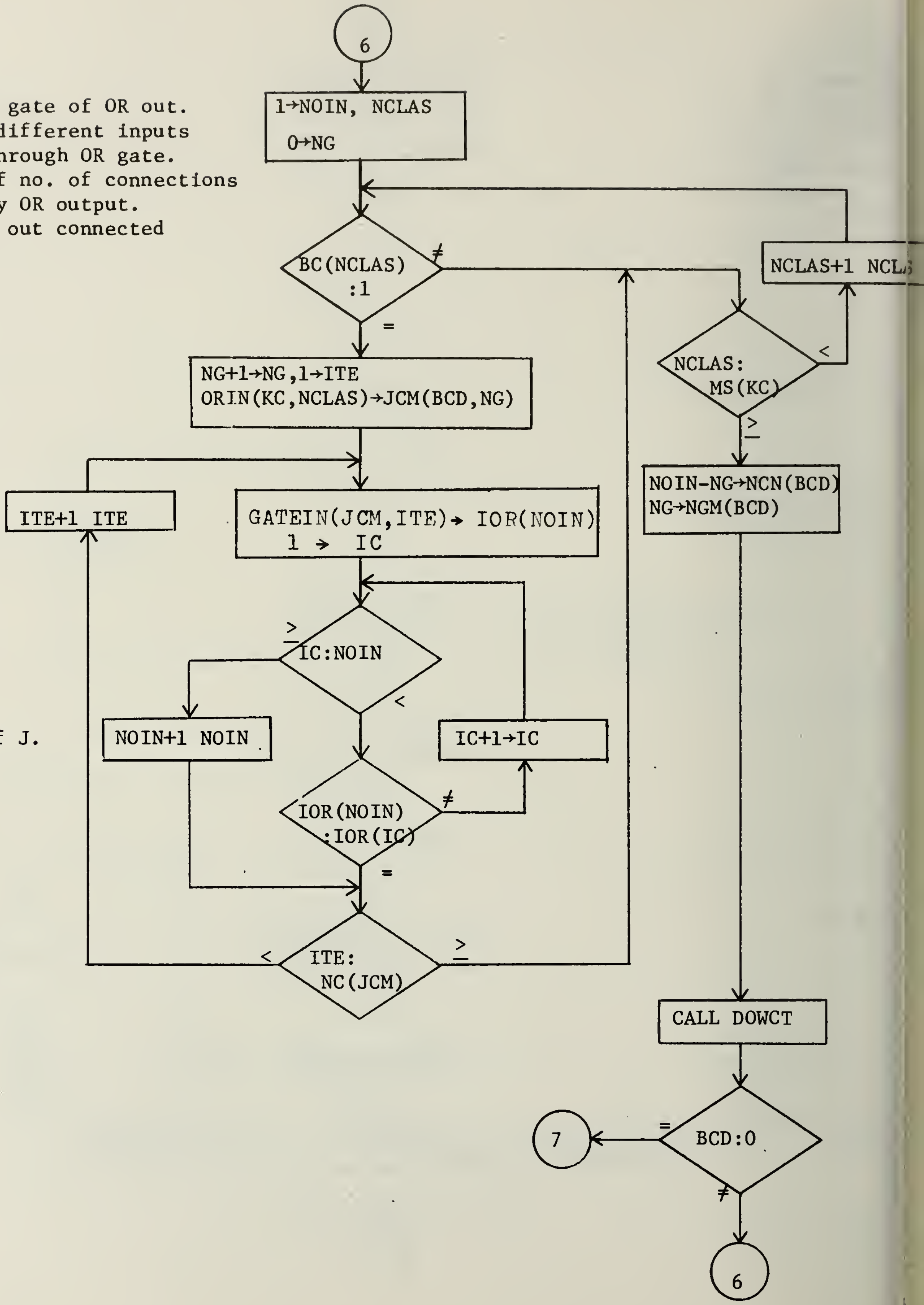
NCLAS: no. of gate of OR out.
NOIN: no. of different inputs
to K through OR gate.
NCN: memory of no. of connections
saved by OR output.
NG: no. of OR out connected
to K.

IOR: inputs of J.

Step 10.

Down-Counter

Step 11.

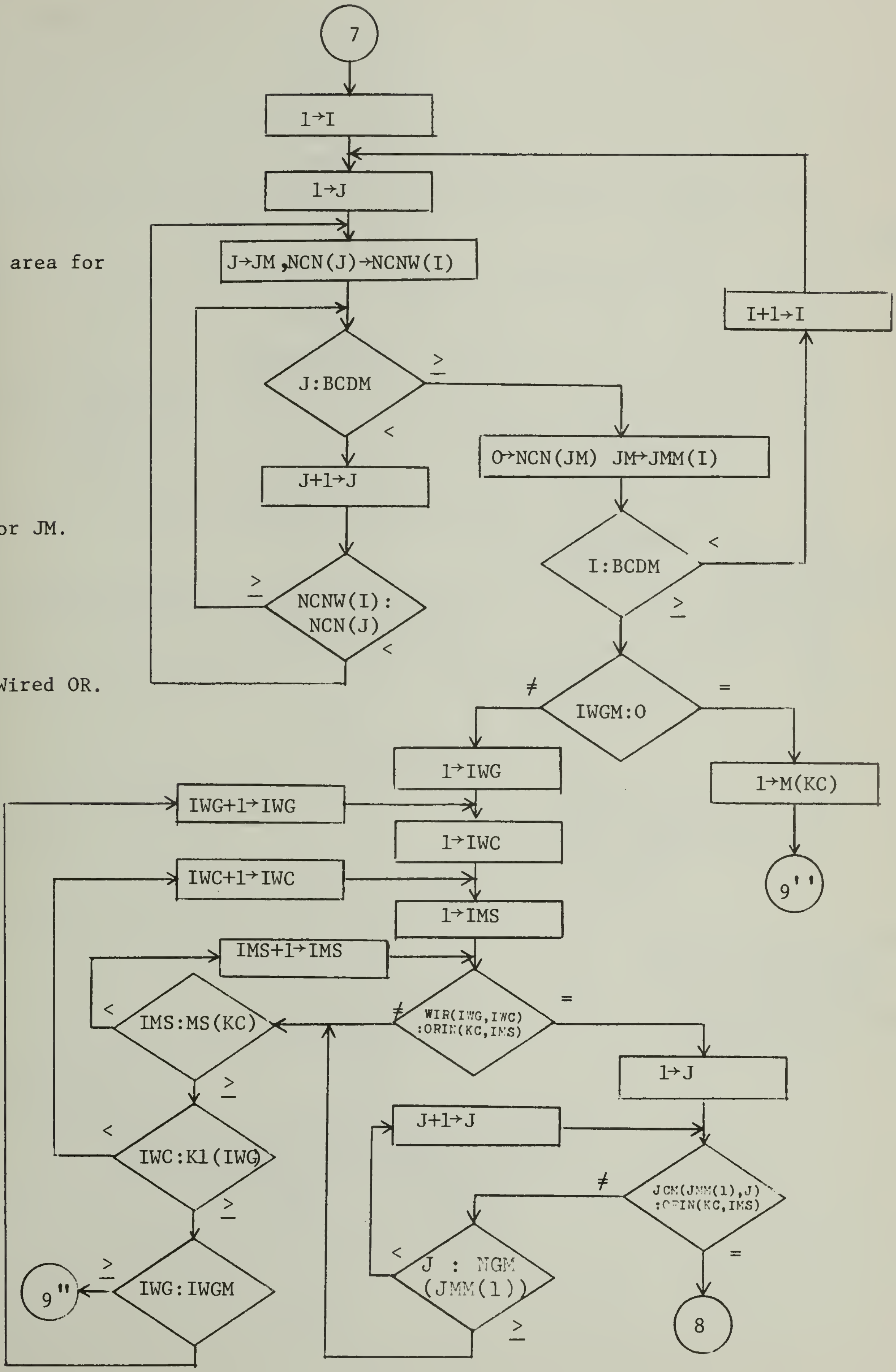


Sorting:

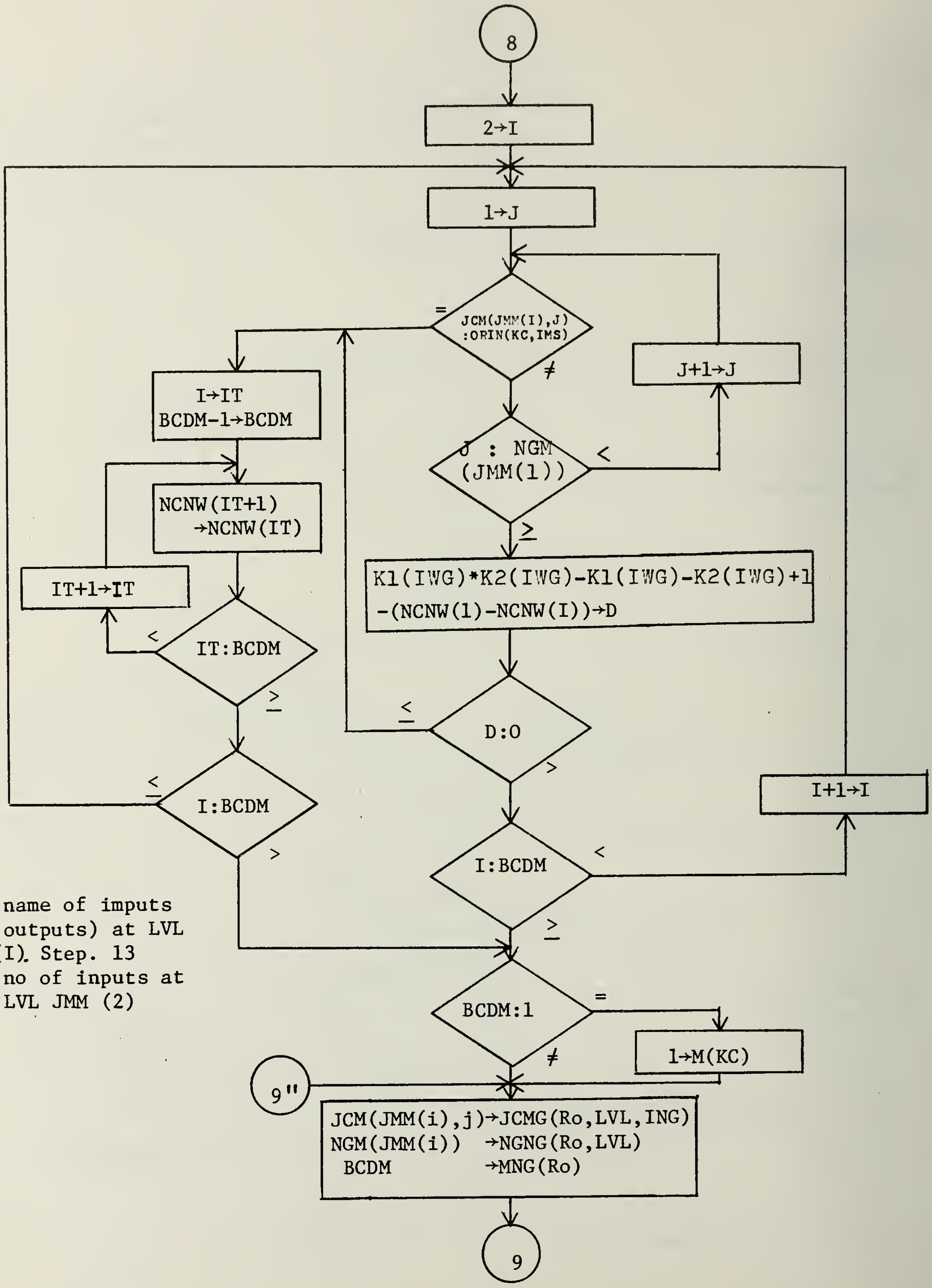
NCNW: Working area for NCN.

JMM: Memory for JM.

IWGM: no. of Wired OR.



Step
12.



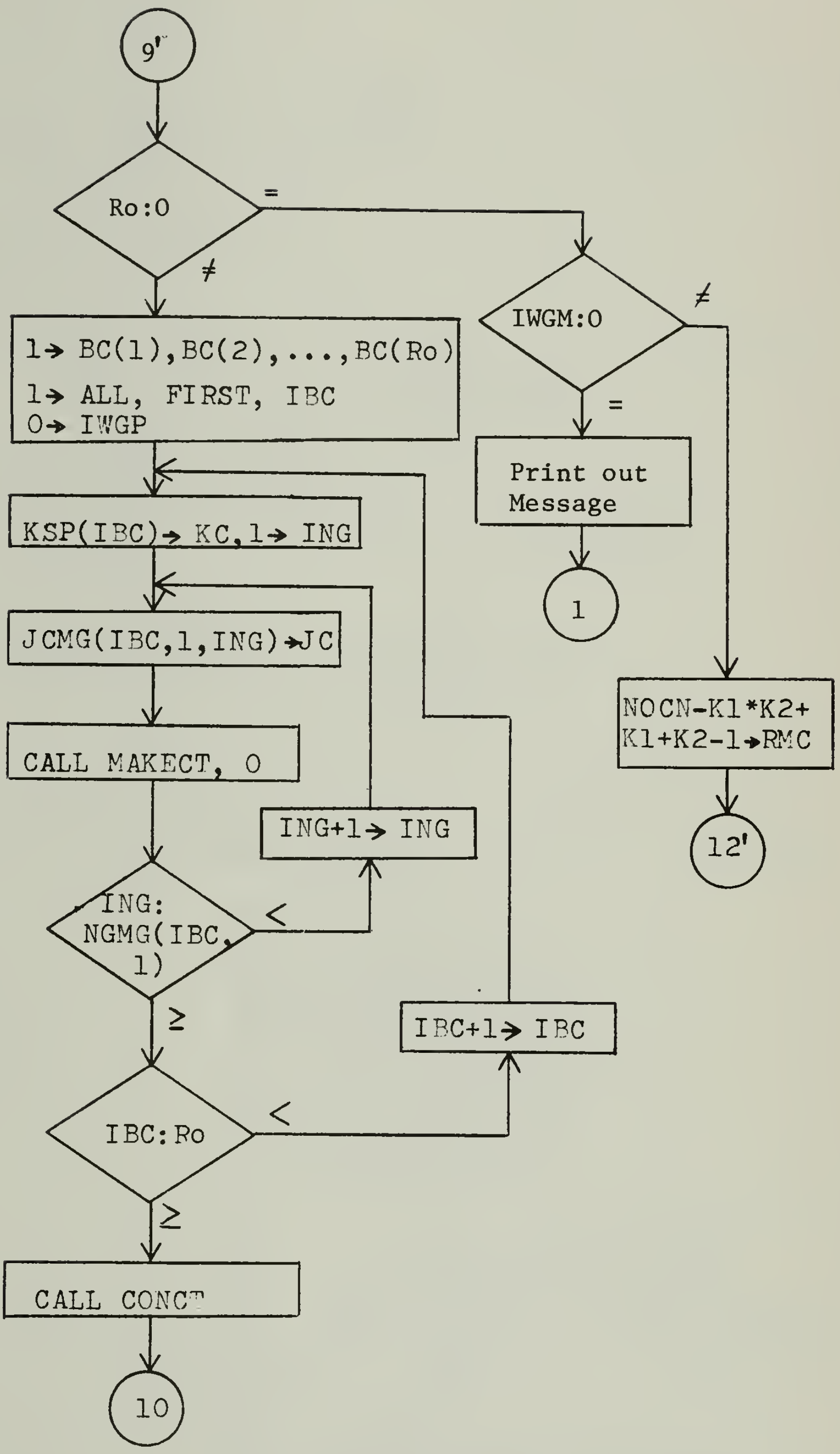
JCM: name of imputs (or outputs) at LVL JMM(I). Step. 13
NGM: no of inputs at LVL JMM (2)

tep 14.

tep 15.

ake a network:

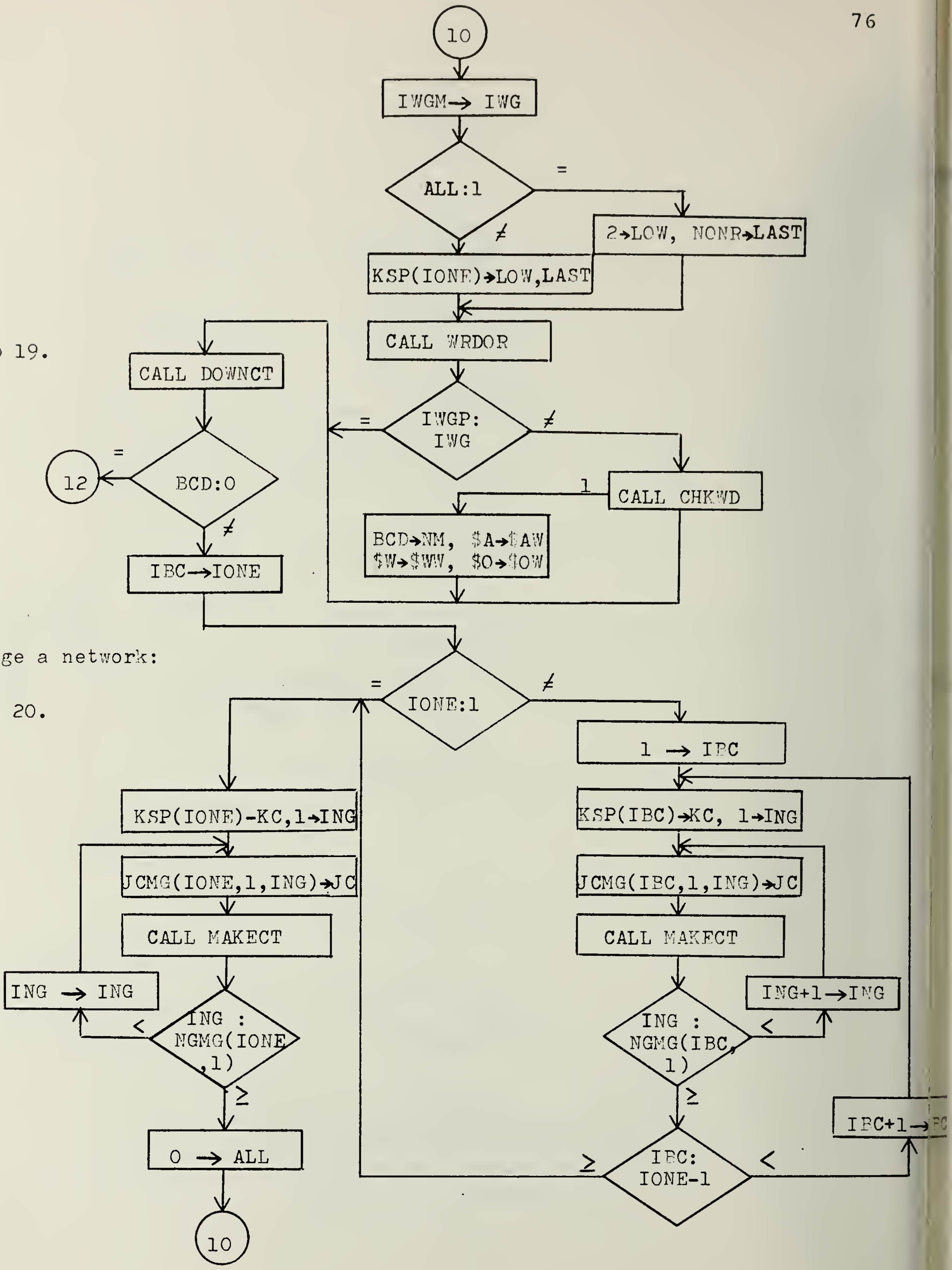
ep 16.



Step 19.

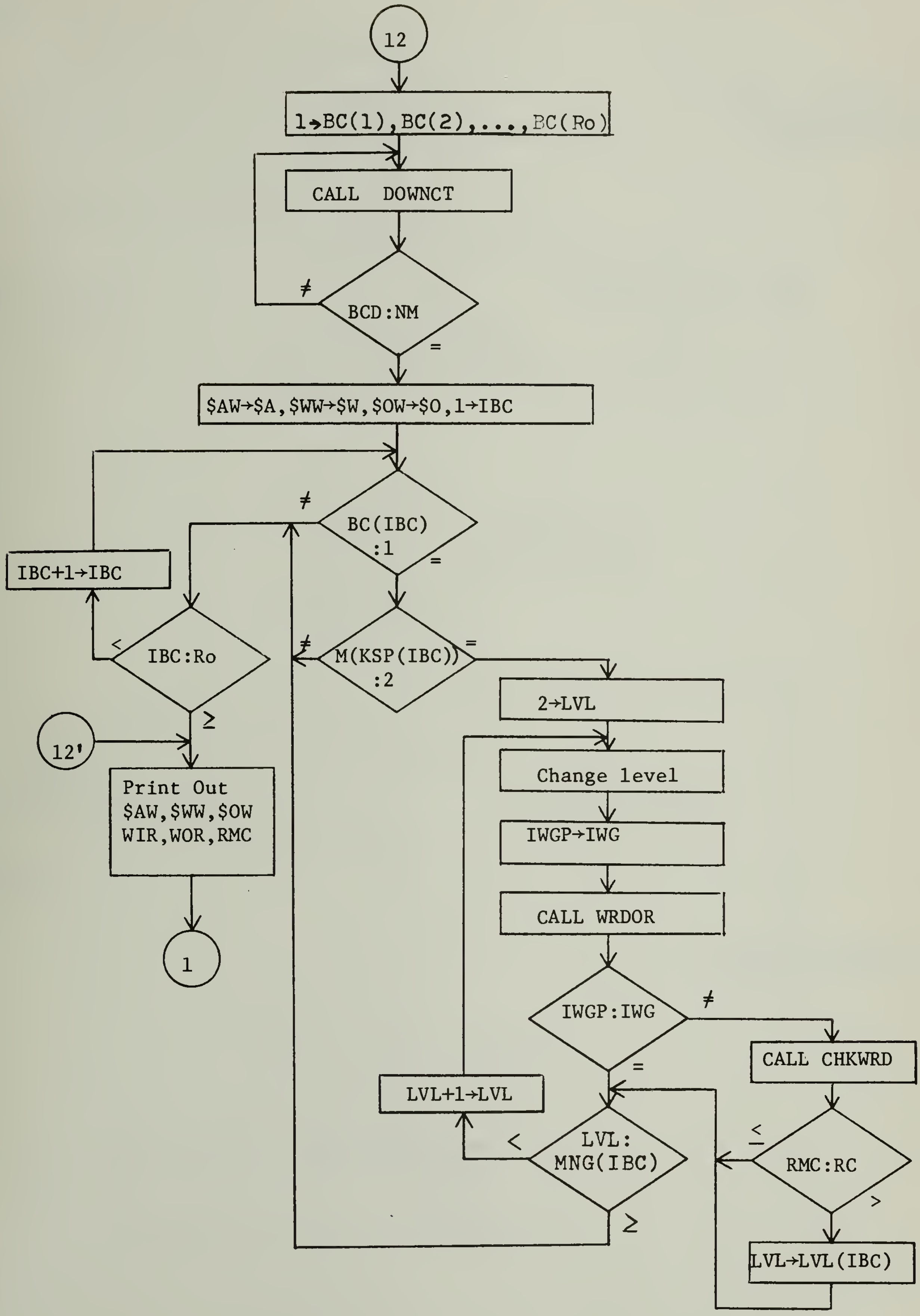
Change a network:

Step 20.



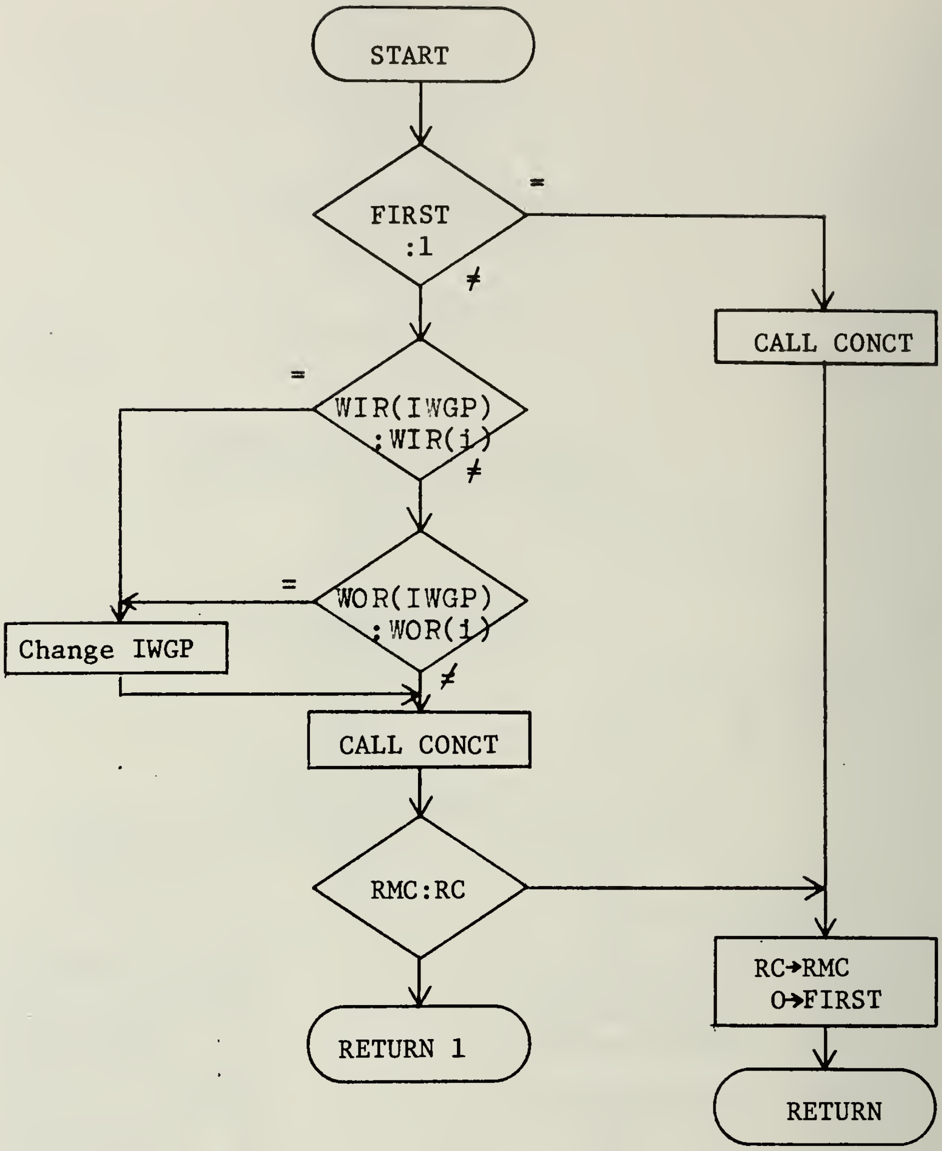
ep 21.

ep 22.

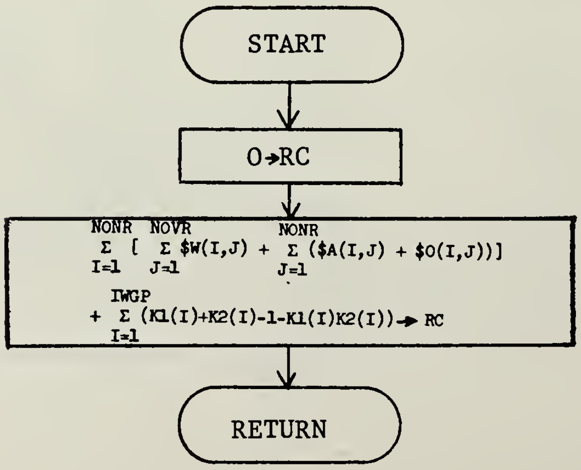


12
1→BC(1),BC(2),...,BC(Ro)
CALL DOWNCT
BCD:NM
≠
=
$AW→$A,$WW→$W,$OW→$O,1→IBC
BC(IBC) :1
≠
=
IBC+1→IBC
IBC:Ro
<
≥
M(KSP(IBC)) :2
≠
=
12'
Print Out $AW,$WW,$OW WIR,WOR,RMC
1
2→LVL
Change level
IWGP→IWG
CALL WRDOR
IWGP:IWG
≠
=
CALL CHKWRD
LVL+1→LVL
LVL: MNG(IBC)
<
≥
RMC:RC
≤
>
LVL→LVL(IBC)

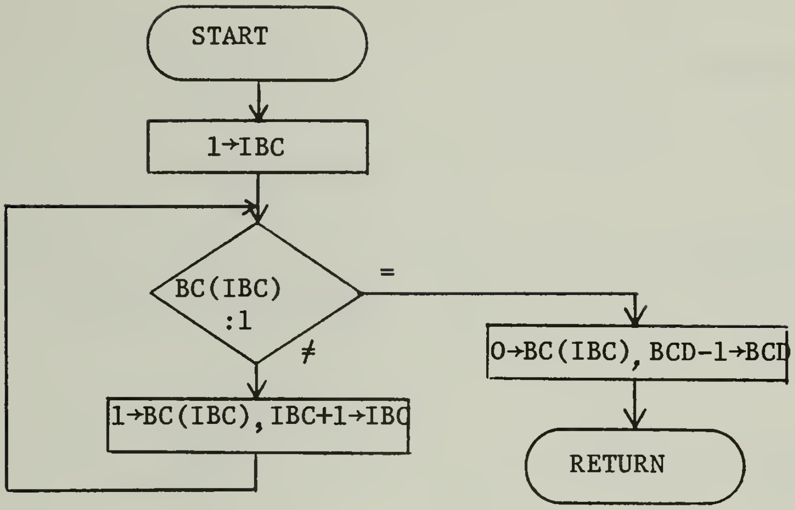
SUBROUTINE CHWKRD
CHECK WIRD-ORs



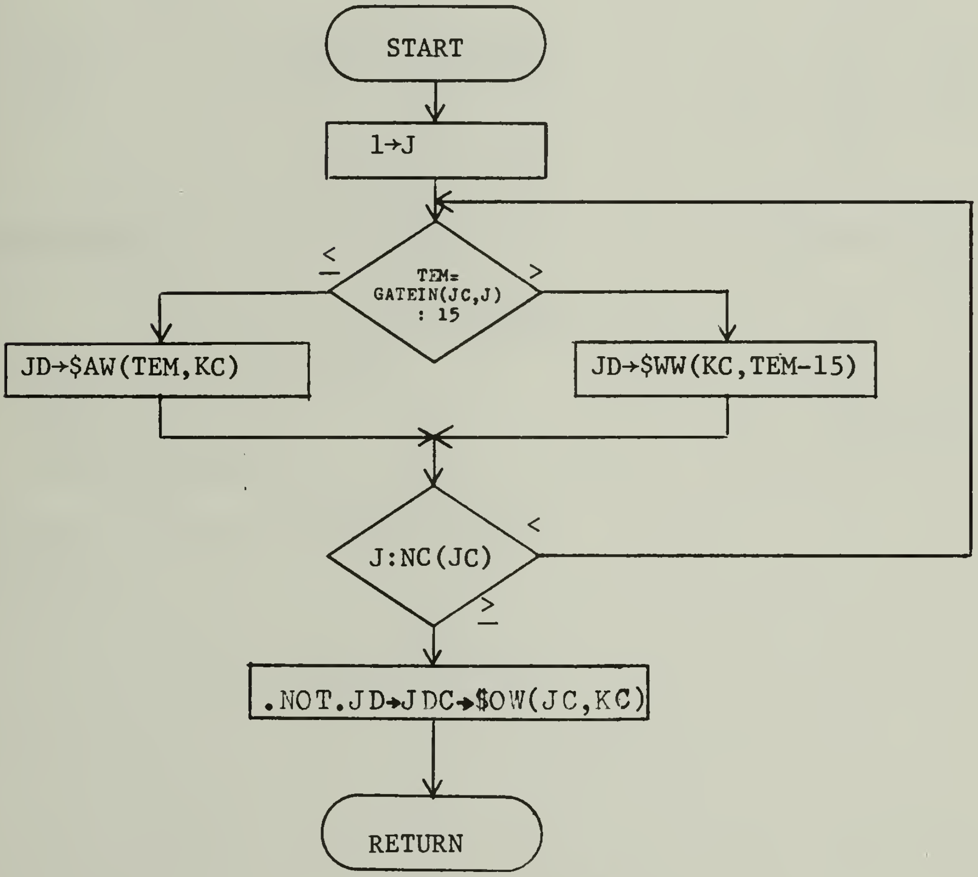
SUBROUTINE CONCT
No. of Connections


START
0→RC
NONR NOVR NONR
Σ [Σ $W(I,J) + Σ ($A(I,J) + $O(I,J))]
I=1 J=1 J=1
IWGP
+ Σ (K1(I)+K2(I)-1-K1(I)K2(I))→ RC
I=1
RETURN

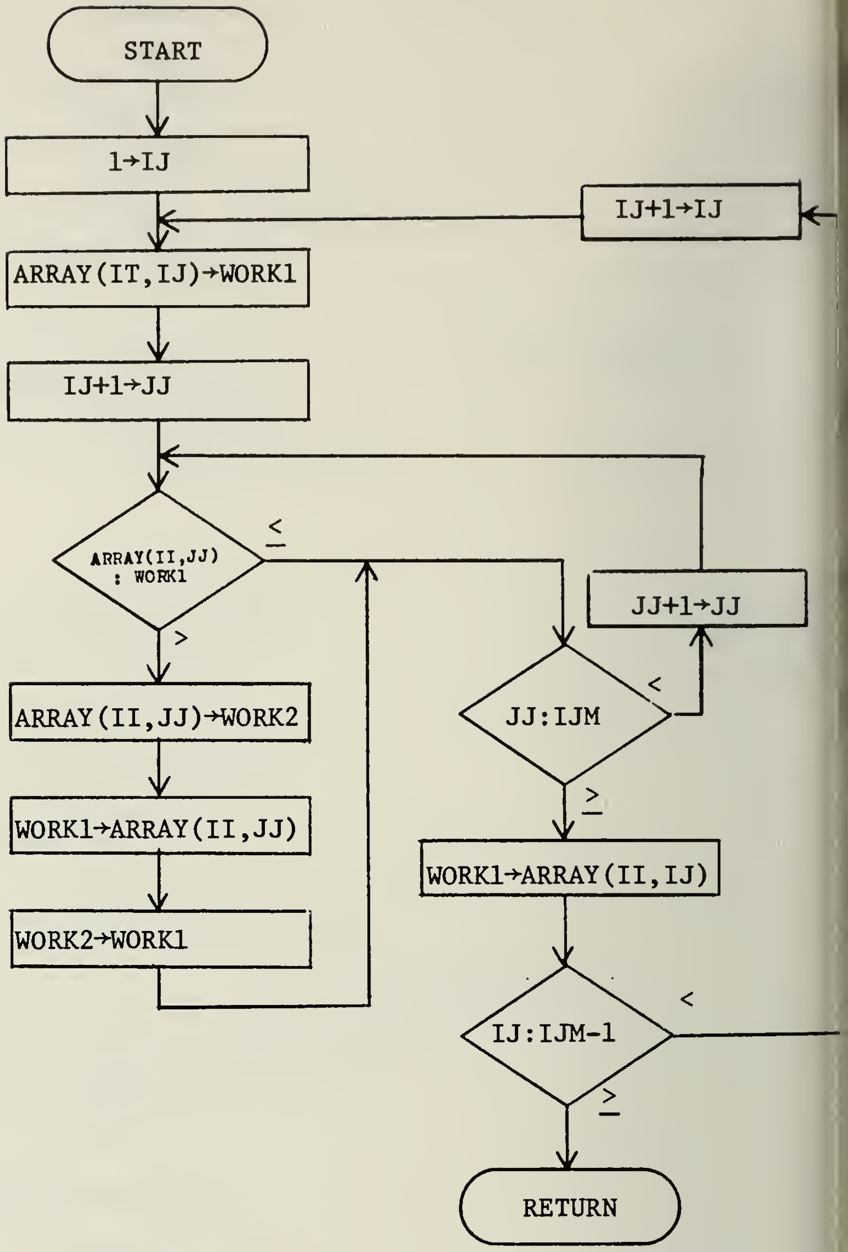
BROUTINE DOWNCT
WN-COUNTER



BROUTINE
KE NETWORK
MAKECT



SUBROUTINE SEQNS
SEQUENCE

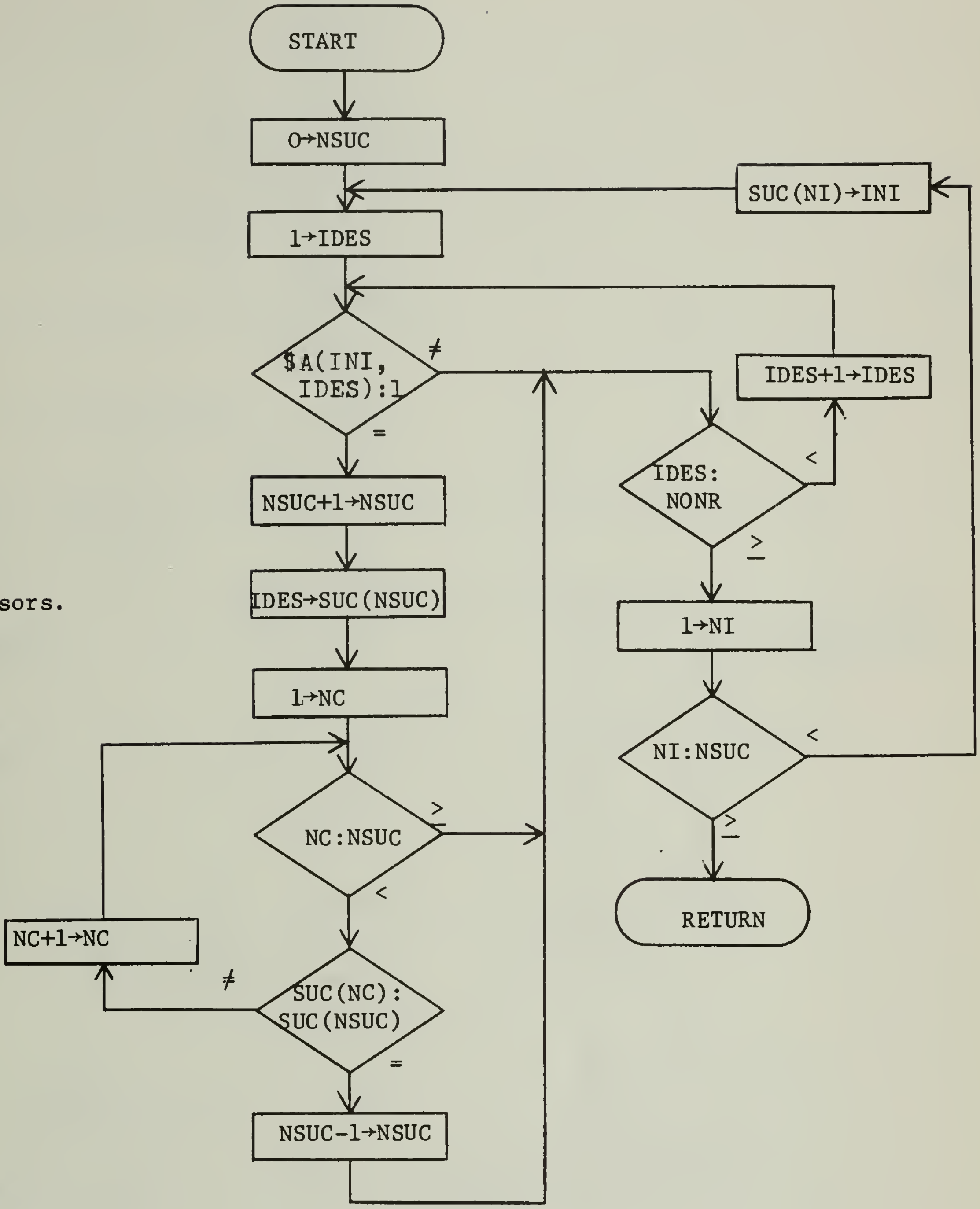


ROUTINE SUCSSR
CESSOR GATES

I: the gate which we want to get its successors.

: successors

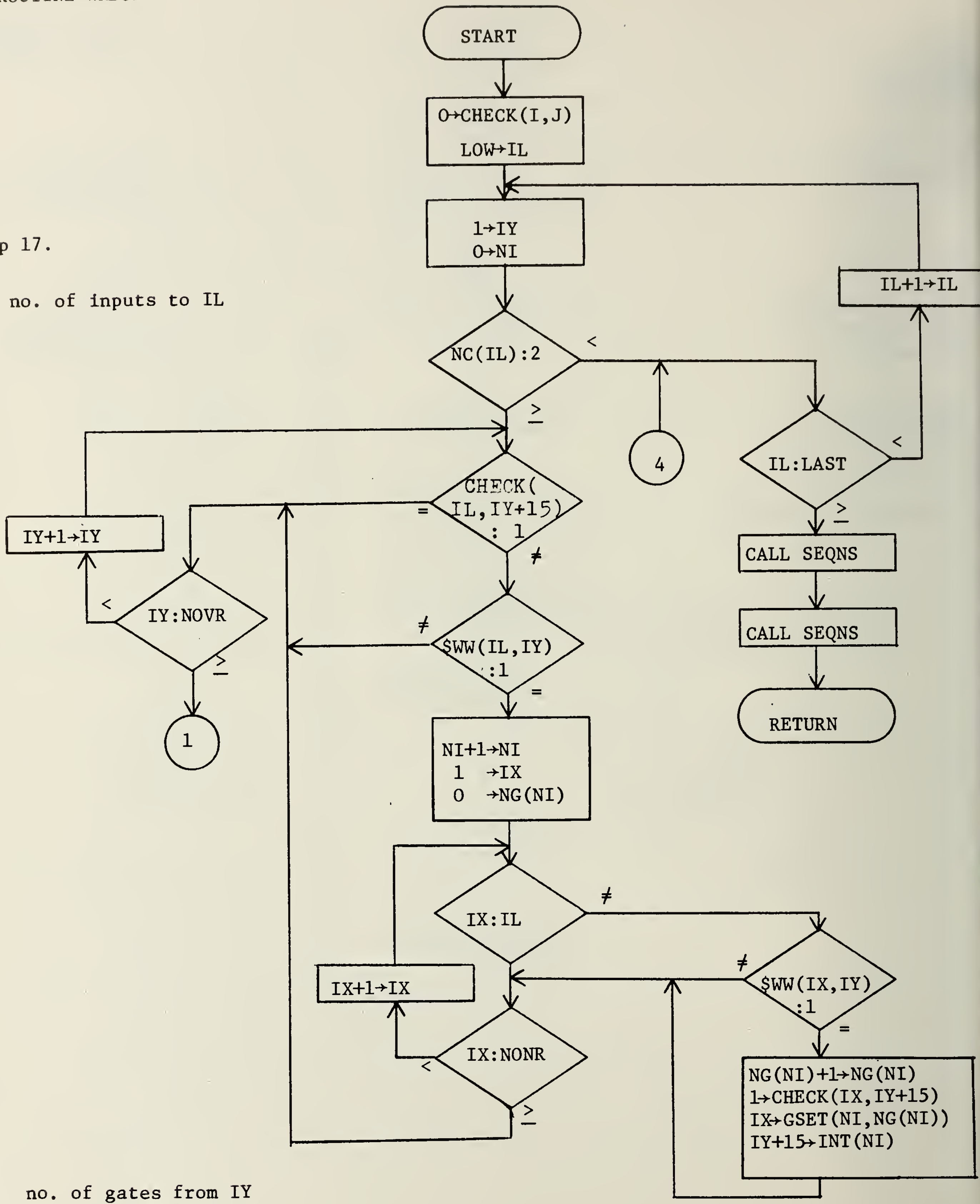
JC: number of successors.



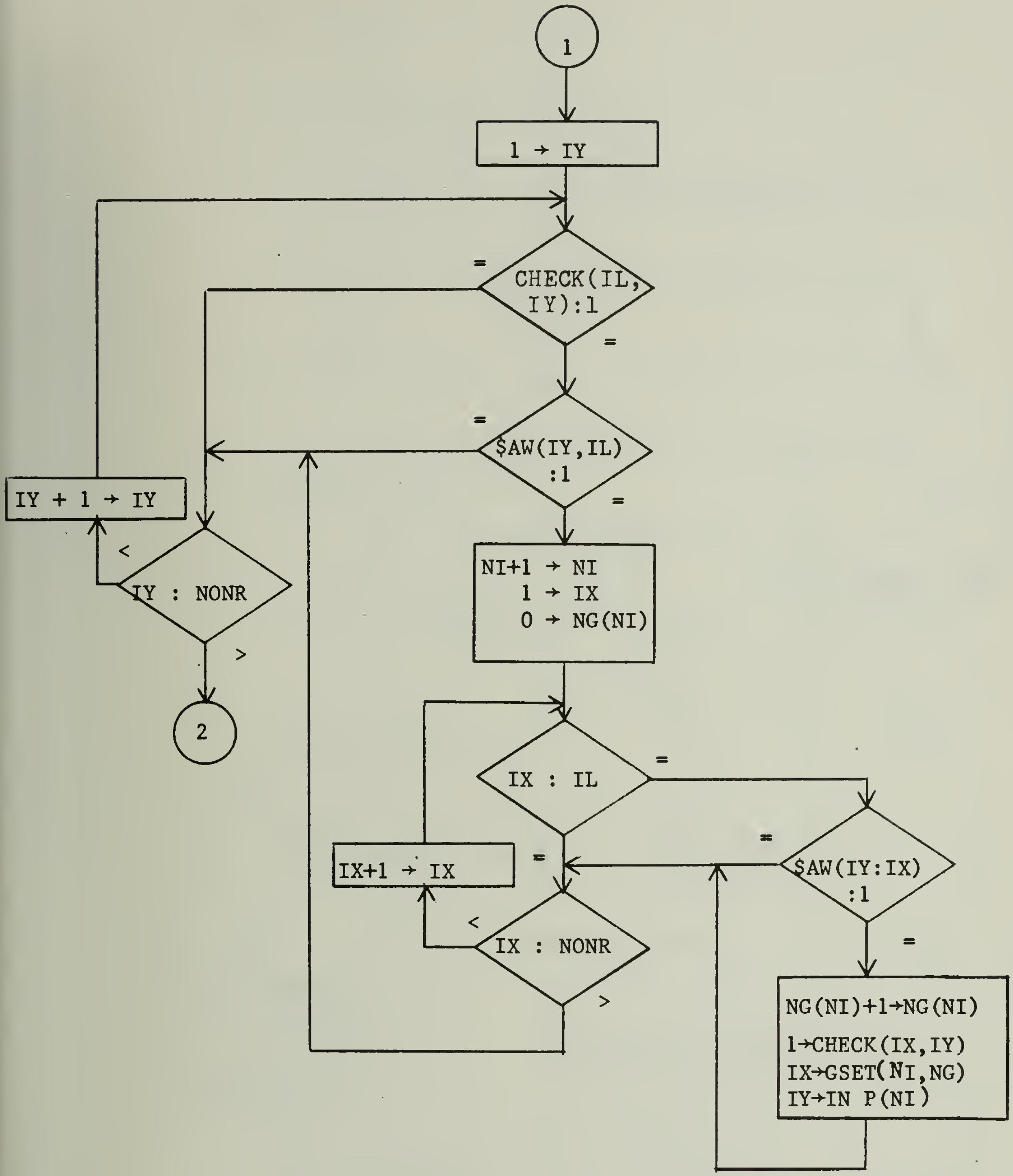
SUBROUTINE WRDOR

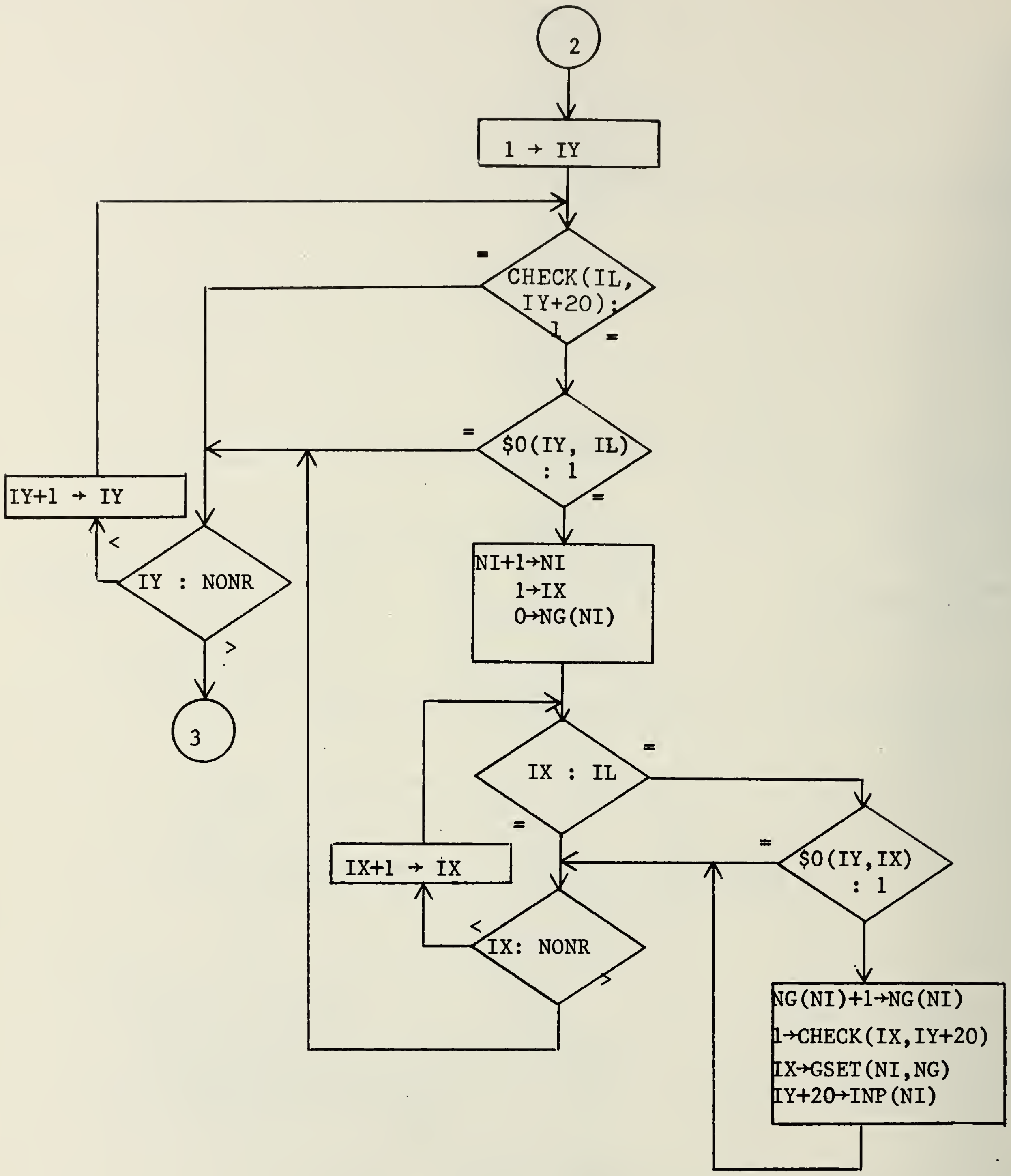
Step 17.

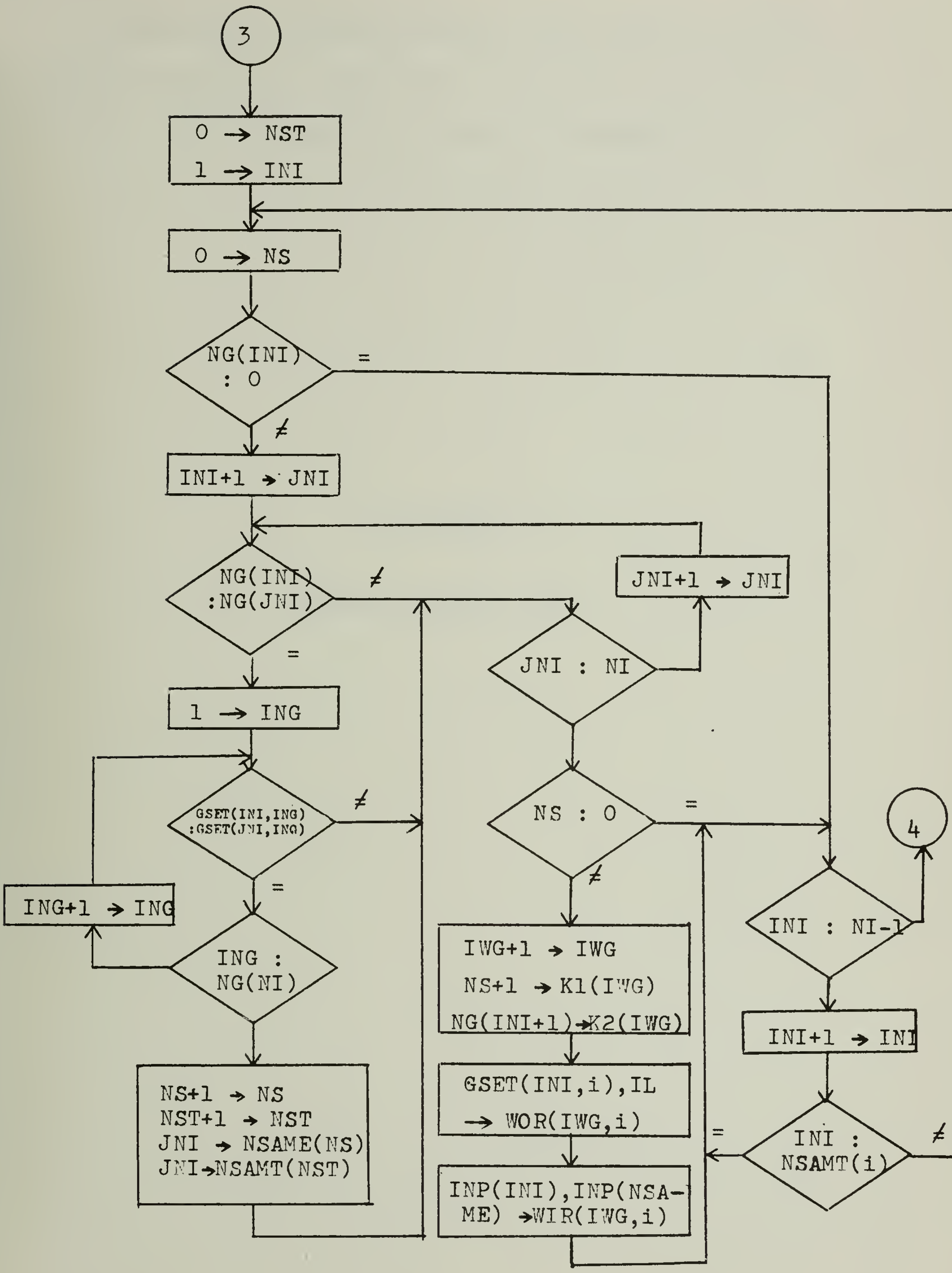
NI: no. of inputs to IL



NG: no. of gates from IY


1
1 → IY
CHECK(IL, IY):1
=
=
$AW(IY,IL) :1
=
=
IY + 1 → IY
<
IY : NONR
>
2
NI+1 → NI
1 → IX
0 → NG(NI)
IX : IL
=
=
$AW(IY:IX) :1
=
=
IX+1 → IX
<
IX : NONR
>
NG(NI)+1→NG(NI)
1→CHECK(IX,IY)
IX→GSET(NI,NG)
IY→IN P(NI)


2
1 → IY
CHECK(IL, IY+20): 1
=
≠
$0(IY, IL) : 1
=
≠
NI+1→NI
1→IX
0→NG(NI)
IY+1 → IY
<
IY : NONR
>
3
IX : IL
=
≠
IX+1 → IX
$0(IY,IX) : 1
=
<
IX: NONR
>
NG(NI)+1→NG(NI)
1→CHECK(IX,IY+20)
IX→GSET(NI,NG)
IY+20→INP(NI)


3
0 → NST
1 → INI
0 → NS
NG(INI) : 0
=
≠
INI+1 → JNI
NG(INI) :NG(JNI)
≠
=
JNI+1 → JNI
JNI : NI
1 → ING
GSET(INI,ING) :GSET(JNI,ING)
≠
=
NS : 0
=
≠
4
ING+1 → ING
ING : NG(NI)
INI : NI-1
IWG+1 → IWG
NS+1 → K1(IWG)
NG(INI+1)→K2(IWG)
INI+1 → INI
NS+1 → NS
NST+1 → NST
JNI → NSAME(NS)
JNI→NSAMT(NST)
GSET(INI,i),IL → WOR(IWG,i)
INI : NSAMT(i)
=
≠
INP(INI),INP(NSAME) →WIR(IWG,i)

APPENDIX D

The Program Package

The complete program package which consists of one main program and seven subroutines as shown in Table 5.1 are shown here. All programs are written in FORTRAN IV.

```
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION $A(15,15),$W(15,4),$O(15,15),$AW(15,15),$WW(15,4),
     1          $OW(15,15),$AR(15,10),$WR(4,10),$OR(15,10),$RA(10,15),
     2          BC(10),IOR(10),JCM(31,10),JMM(31),KSP(10),LVLM(5),M(15),
     3          MNG(5),MS(15),NCN(31),NCNW(31),NGM(31),NGMG(5,15),
     4          ORIN(15,5),SUC(14),WIR(10,10),WOR(10,10),JCMG(5,5,5)
    1 FORMAT(2X,I1,2X,60I1)
    2 FORMAT(21X,I2,6X,I3,7X,I3)
    3 FORMAT(5X,75I1)
    4 FORMAT(//,' NO.',I2,'.  FCT=',I3,'  COMP=',I3,'  NOVR=',I1,
     1      '  NONR=',I2,'  NOCN=',I3,/,'     ORIGINAL TABLE')
    5 FORMAT(' W  ',60I1)
    6 FORMAT(' A  ',75I1)
    7 FORMAT(' NO OR OUTPUT, NO WIRED OR AVAILABLE')
    8 FORMAT( /,'     NEW TABLE')
    9 FORMAT(' O  ',75I1)
   10 FORMAT(' WR ',40I1)
   11 FORMAT(' AR ',75I1)
   12 FORMAT(' OR ',75I1)
   13 FORMAT(' RA ',75I1)
   14 FORMAT(' MIN. NO. OF CONNECTIONS=',I3)
C    READ DATA CARDS
  100 READ 1,D2,$W
      IF(D2.GT.5) STOP
      IF(D2-3) 115,110,105
  105 NOVR=$W(2,1)
      NONR=10*$W(5,1)+$W(6,1)
      GO TO 100
  110 READ 2,NO,FCT,COMP
      GO TO 100
  115 READ 3,(($A(I,J),I= 1, 5),J=1,15)
      READ 3,(($A(I,J),I= 6,10),J=1,15)
      READ 3,(($A(I,J),I=11,15),J=1,15)
      DO 120 I=1,15
      NC(I)=0
      DO 120 J=1,5
  120 GATEIN(I,J)=0
C    COUNT THE NUMBER OF CONNECTIONS
      DO 150 I=1,NONR
      DO 150 J=1,NOVR
      IF($W(I,J).EQ.0) GO TO 150
      NC(I)=NC(I)+1
      GATEIN(I,NC(I))=J+15
  150 CONTINUE
      DO 220 I=1,NONR
      DO 220 J=1,NONR
      IF($A(I,J).EQ.0) GO TO 220
      NC(J)=NC(J)+1
      GATEIN(J,NC(J))=I
  220 CONTINUE
      NOCN=0
      DO 230 I=1,NONR
  230 NOCN=NOCN+NC(I)
      PRINT 4,NO,FCT,COMP,NOVR,NONR,NOCN
      PRINT 5,$W
      PRINT 6,$A
      IF(NONR.GE.3) GO TO 300
  250 PRINT 7
      GO TO 100
C    INITIALIZATION
```

```
 300  DO 310 I=1,15
      MS(I)=0
      DO 305 J=1,5
 305  ORIN(I,J)=0
      DO 308 J=1,4
 308  $WW(I,J)=$W(I,J)
      DO 310 J=1,15
      $O(I,J)=0
      $OW(I,J)=0
 310  $AW(I,J)=$A(I,J)
      RO=0
C     FIND OR OUTPUTS AVAILABLE
      DO 500 KC=1,NONR
      IF(NC(KC).LT.2) GO TO 500
      CALL SUCSSR($A,KC,SUC,NSUC)
      DO 490 JC=1,NONR
      IF(JC.EQ.KC) GO TO 490
      IF(NSUC.EQ.0) GO TO 330
      DO 320 IS=1,NSUC
      IF(JC.EQ.SUC(IS)) GO TO 490
 320  CONTINUE
 330  IF($A(JC,KC).EQ.1) GO TO 490
C     FIND SUBSET
      INJ=1
 340  INK=1
 350  IF(GATEIN(JC,INJ).EQ.GATEIN(KC,INK)) GO TO 360
      IF(INK.GE.NC(KC)) GO TO 490
      INK=INK+1
      GO TO 350
 360  IF(INJ.GE.NC(JC)) GO TO 400
      INJ=INJ+1
      GO TO 340
C     MAKE NETWORK WITH NOR-OR OUTPUT
 400  MS(KC)=MS(KC)+1
      CALL MAKECT($AW,$WW,$OW,JC,KC,0)
      ORIN(KC,MS(KC))=JC
 490  CONTINUE
 500  CONTINUE
C     CHECK WIRED OR AVAILABLE
      IWGM=0
      CALL WRDOR($AW,$WW,$OW,2,NONR,WIR,WOR,IWGM)
      DO 900 KC=1,NONR
      IF(MS(KC).LT.1) GO TO 900
      RO=RO+1
      M(RO)=0
      KSP(RO)=KC
      IF(MS(KC).LT.2) GO TO 900
      M(RO)=2
C     GATES IN GROUP M
      ILT=MS(KC)
      DO 530 I=1,ILT
 530  BC(I)=1
      IFT=ILT+1
      DO 532 I=IFT,10
 532  BC(I)=0
      BCD=2**MS(KC)-1
      BCDM=BCD
C     COUNT NO. OF CONNECTIONS IN ALL LEVELS TO GATE K
 600  NOIN=1
      NG=0
      DO 650 NCLAS=1,ILT
      IF(BC(NCLAS).NE.1) GO TO 650
      NG=NG+1
      JCM(BCD,NG)=ORIN(KC,NCLAS)
      ILS=NC(JCM(BCD,NG))
      DO 640 ITE=1,ILS
      IOR(NOIN)=GATEIN(JCM(BCD,NG),ITE)
      IC=1
 620  IF(IC.GE.NOIN) GO TO 630
      IF(IOR(NOIN).EQ.IOR(IC)) GO TO 640
      IC=IC+1
      GO TO 620
 630  NOIN=NOIN+1
 640  CONTINUE
 650  CONTINUE
      NCN(BCD)=NOIN-NG-1
      NGM(BCD)=NG
      CALL DOWNCT(BC,BCD,IBC)
      IF(BCD.NE.0) GO TO 600
C     SORTING(NO. OF CONNECTION TO GATE K)
      DO 730 I=1,BCDM
      J=1
 705  JM=J
      NCNW(I)=NCN(J)
 710  IF(J.GE.BCDM) GO TO 720
      J=J+1
      IF(NCNW(I).GE.NCN(J)) GO TO 710
      GO TO 705
 720  NCN(JM)=0
 730  JMM(I)=JM
      IF(IWGM.NE.0) GO TO 740
      M(RO)=1
      BCDM=1
      GO TO 890
C     CONSIDER WIRED OR AVAILABLE
 740  DO 770 IWG=1,IWGM
      ILS=K1(IWG)
      DO 770 IWC=1,ILS
      DO 770 IMS=1,ILT
      IF(WIR(IWG,IWC).NE.ORIN(KC,IMS)) GO TO 770
      ILR=NGM(JMM(1))
      DO 760 J=1,ILR
      IF(JCM(JMM(1),J).EQ.ORIN(KC,IMS)) GO TO 800
 760  CONTINUE
 770  CONTINUE
      GO TO 890
C     GET NO. OF CONNECTIONS THAT CAN BE SAVED BY THIS CONNECTION
 800  I=2
 810  ILR=NGM(JMM(I))
      DO 820 J=1,ILR
      IF(JCM(JMM(I),J).EQ.ORIN(KC,IMS)) GO TO 830
 820  CONTINUE
      IF(K1(IWG)*K2(IWG)-K1(IWG)-K2(IWG)+1-NCNW(1)+NCNW(I).LE.0)GOTO 830
      IF(I.GE.BCDM) GO TO 840
      I=I+1
      GO TO 810
 830  BCDM=BCDM-1
      DO 835 IT=I,BCDM
 835  NCNW(IT)=NCNW(IT+1)
      IF(I.LE.BCDM) GO TO 810
 840  IF(BCDM.NE.1) GO TO 890
      M(RO)=1
 890   DO 895 LVL=1,BCDM
       NGMG(RO,LVL)=NGM(JMM(LVL))
       ILR=NGM(JMM(LVL))
       DO 895 ING=1,ILR
 895   JCMG(RO,LVL,ING)=JCM(JMM(LVL),ING)
       MNG(RO)=BCDM
 900   CONTINUE
C     WIRED ORS
C     MAKE NETWORK
       IF(RO.NE.0) GO TO 904
       IF(IWGM.EQ.0) GO TO 250
       RMC=NCCN
       DO 903 I=1,IWGM
 903   RMC=RMC-K1(I)*K2(I)+K1(I)+K2(I)-1
       GO TO 1270
 904   DO 905 I=1,RO
 905   BC(I)=1
       IFT=RO+1
       DO 910 I=IFT,10
 910   BC(I)=0
       BCD=2**RO-1
       ALL=1
       FIRST=1
       IONE=0
       IWGM=0
       DO 990 IBC=1,RO
       IF(M(IBC).GE.1) GO TO 980
       CALL MAKECT($A,$W,$O,ORIN(KSP(IBC),1),KSP(IBC),0)
       GO TO 990
 980   ILR=NGMG(IBC,1)
       DO 985 ING=1,ILR
 985   CALL MAKECT($A,$W,$O,JCMG(IBC,1,ING),KSP(IBC),0)
 990   CONTINUE
       CALL CCNCT($A,$W,$O,0,ROR)
C     MAKE WIRED ORS
1000   IWG=IWGM
       IF(ALL.EQ.1) GO TO 1020
       CALL WRDOR($A,$W,$O,KSP(IONE),KSP(IONE),WIR,WOR,IWGM)
       GO TO 1030
1020   CALL WRDOR($A,$W,$O,2,NONR,WIR,WOR,IWGM)
1030   IF(IWGM.EQ.IWG) GO TO 1090
       CALL CHKWRD($A,$W,$O,WIR,WOR,IWG,IWGM,RMC,FIRST,&1090)
       NM=BCD
       DO 1085 LC=1,NONR
       DO 1080 LD=1,NOVR
1080   $WW(LC,LD)=$W(LC,LD)
       DO 1085 LD=1,NONR
       $AW(LC,LD)=$A(LC,LD)
1085   $OW(LC,LD)=$O(LC,LD)
1090   IF(BCD.EQ.0) GO TO 1200
       CALL DOWNCT(BC,BCD,IONE)
C     CHANGE NETWORK
       IF(IONE.EQ.1) GO TO 1150
       ILS=IONE-1
       DO 1130 IBC=1,ILS
       IF(M(IBC).GE.1) GO TO 1120
       CALL MAKECT($A,$W,$O,ORIN(KSP(IBC),1),KSP(IBC),0)
       GO TO 1130
1120   ILR=NGMG(IBC,1)
       DO 1125 ING=1,ILR
1125   CALL MAKECT($A,$W,$O,JCMG(IBC,1,ING),KSP(IBC),0)
```

```
1130  CONTINUE
1150  IF(M(IONE).GE.1) GO TO 1155
      CALL MAKECT($A,$W,$O,ORIN(KSP(IONE),1),KSP(IONE),1)
      GO TO 1165
1155  ILR=NGMG(IONE,1)
      DO 1160 ING=1,ILR
1160  CALL MAKECT($A,$W,$O,JCMG(IONE,1,ING),KSP(IONE),1)
1165  ALL=0
      GO TO 1000
C     CHANGE LEVEL IN CONNECTION GROUP M
1200  IF(IWGM.EQ.0) GO TO 1282
      IF(RMC.GE.ROR) GO TO 1282
      DO 1201 I=1,RO
1201  LVLM(I)=1
      DO 1205 I=1,RO
1205  BC(I)=1
      DO 1210 I=IFT,10
1210  BC(I)=0
      BCD=2**RO-1
1213  IF(BCD.EQ.NM) GO TO 1215
      CALL DCWNCT(BC,BCD,IBC)
      GO TO 1213
1215  DO 1250 IBC=1,RO
      IF(BC(IBC).NE.1) GO TO 1250
      IF(M(IBC).NE.2) GO TO 1250
      DO 1217 I=1,NONR
      DO 1216 J=1,NOVR
1216  $W(I,J)=$WW(I,J)
      DO 1217 J=1,NONR
      $A(I,J)=$AW(I,J)
1217  $O(I,J)=$OW(I,J)
      ILS=MNG(IBC)
      DO 1240 LVL=2,ILS
      ILR=NGMG(IBC,LVL-1)
      DO 1220 ING=1,ILR
1220  CALL MAKECT($A,$W,$O,JCMG(IBC,LVL-1,ING),KSP(IBC),1)
      ILR=NGMG(IBC,LVL)
      DO 1230 ING=1,ILR
1230  CALL MAKECT($A,$W,$O,JCMG(IBC,LVL,ING),KSP(IBC),0)
      IWG=IWGM
      CALL WRDOR($A,$W,$O,KSP(IBC),KSP(IBC),WIR,WOR,IWGM)
      IF(IWGM.EQ.IWG) GO TO 1240
      CALL CHKWRD($A,$W,$O,WIR,WOR,IWG,IWGM,RMC,FIRST,&1240)
      LVLM(IBC)=LVL
1240  CONTINUE
1250  CONTINUE
C     GET FINAL NETWORK
      DO 1265 IBC=1,RO
      IF(LVLM(IBC).EQ.1) GO TO 1265
      ILR=NGMG(IBC,1)
      DO 1255 ING=1,ILR
1255  CALL MAKECT($AW,$WW,$OW,JCMG(IBC,1,ING),KSP(IBC),1)
      ILR=NGMG(IBC,LVLM(IBC))
      DO 1260 ING=1,ILR
1260  CALL MAKECT($AW,$WW,$OW,JCMG(IBC,LVLM(IBC),ING),KSP(IBC),0)
1265  CONTINUE
      IWGM=0
      CALL WRDOR($AW,$WW,$OW,2,NONR,WIR,WOR,IWGM)
      CALL CONCT($AW,$WW,$OW,IWGM,RMC)
1270  DO 1272 I=1,10
      DO 1271 J=1,4
```

```
1271  $WR(J,I)=0
      DO 1272 J=1,15
      $AR(J,I)=0
      $OR(J,I)=0
1272  $RA(I,J)=0
      DO 1280 I=1,IWGM
      ILT=K1(I)
      ILS=K2(I)
      DO 1278 J=1,ILT
      IF(WIR(I,J).GT.20) GO TO 1276
      IF(WIR(I,J).GT.15) GO TO 1274
      $AR(WIR(I,J),I)=1
      DO 1273 JO=1,ILS
1273  $AW(WIR(I,J),WOR(I,JO))=0
      GO TO 1278
1274  $WR(WIR(I,J)-15,I)=1
      DO 1275 JO=1,ILS
1275  $WW(WOR(I,JO),WIR(I,J)-15)=0
      GO TO 1278
1276  $OR(WIR(I,J)-20,I)=1
      DO 1277 JO=1,ILS
1277  $OW(WIR(I,J)-20,WOR(I,JO))=0
1278  CONTINUE
      DO 1280 J=1,ILS
1280  $RA(I,WOR(I,J))=1
      GO TO 1290
1282  DO 1286 I=1,RO
      IF(M(I).GE.1) GO TO 1284
      CALL MAKECT($A,$W,$O,ORIN(KSP(I),1),KSP(I),0)
      GO TO 1286
1284  ILR=NGMG(I,1)
      DO 1285 ING=1,ILR
1285  CALL MAKECT($A,$W,$O,JCMG(I,1,ING),KSP(I),0)
1286  CONTINUE
      RMC=ROR
      IWGM=0
      DO 1288 I=1,NONR
      DO 1287 J=1,NOVR
1287  $WW(I,J)=$W(I,J)
      DO 1288 J=1,NONR
      $AW(I,J)=$A(I,J)
1288  $OW(I,J)=$O(I,J)
1290  PRINT 8
      PRINT 5,$WW
      PRINT 6,$AW
      IF(RO.EQ.0) GO TO 1291
      PRINT 9,$OW
      IF(IWGM.EQ.0) GO TO 1295
1291  PRINT 10,$WR
      PRINT 11,$AR
      IF(RO.EQ.0) GO TO 1292
      PRINT 12,$OR
1292  PRINT 13,$RA
1295  PRINT 14,RMC
      GO TO 100
      END
      SUBROUTINE CHKWRD($A,$W,$O,WIR,WOR,IWG,IWGP,RMC,F,*)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION $A(15,15),$W(15,4),$O(15,15),WIR(10,10),WOR(10,10)
      IF(F.EQ.1) GO TO 90
      I=IWG+1
 5    DO 40 J=1,IWG
      IF(K1(I).NE.K1(J)) GO TO 40
      ILT=K1(I)
      DO 10 IW=1,ILT
      IF(WIR(I,IW).NE.WIR(J,IW)) GO TO 40
10    CONTINUE
      IWGP=IWGP-1
      K2(J)=K2(J)+1
      ILT=K2(J)-1
      DO 15 IW=1,ILT
      IF(WOR(J,IW).NE.WOR(I,IW)) GO TO 20
15    CONTINUE
      WOR(J,K2(J))=WOR(I,K2(J))
      GO TO 25
20    WOR(J,K2(J))=WOR(I,IW)
25    CALL SEQNS(WOR,J,K2(J))
      DO 35 LC=I,IWGP
      K1(LC)=K1(LC+1)
      K2(LC)=K2(LC+1)
      ILT=K1(LC)
      DO 30 LD=1,ILT
30    WIR(LC,LD)=WIR(LC+1,LD)
      ILT=K2(LC)
      DO 35 LD=1,ILT
35    WOR(LC,LD)=WOR(LC+1,LD)
      GO TO 82
40    CONTINUE
      DO 80 J=1,IWG
      IF(K2(I).NE.K2(J)) GO TO 80
      ILT=K2(I)
      DO 45 IW=1,ILT
      IF(WOR(I,IW).NE.WOR(J,IW)) GO TO 80
45    CONTINUE
      IWGP=IWGP-1
      K1(J)=K1(J)+1
      ILT=K1(J)-1
      DO 50 IW=1,ILT
      IF(WIR(J,IW).NE.WIR(I,IW)) GO TO 55
50    CONTINUE
      WIR(J,K2(J))=WIR(I,K2(J))
      GO TO 60
55    WIR(J,K2(J))=WIR(I,IW)
60    CALL SEQNS(WIR,J,K2(J))
      DO 65 LC=I,IWGP
      K1(LC)=K1(LC+1)
      K2(LC)=K2(LC+1)
      ILT=K1(LC)
      DO 65 LD=1,ILT
65    WIR(LC,LD)=WIR(LC+1,LD)
      ILT=K2(LC)
      DO 70 LD=1,ILT
70    WOR(LC,LD)=WOR(LC+1,LD)
      GO TO 82
80    CONTINUE
      IF(I.GE.IWGP) GO TO 85
      I=I+1
      GO TO 5
82    IF(I.GT.IWGP) GO TO 85
      GO TO 5
85    CALL CONCT($A,$W,$O,IWGP,RC)
```

```
      IF(RMC.LE.RC) RETURN1
      GO TO 95
90    CALL CONCT($A,$W,$O,IWGP,RC)
95    RMC=RC
      F=0
      RETURN
      END
      SUBROUTINE CONCT($A,$W,$O,IWGP,RC)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION $A(15,15),$W(15,4),$O(15,15)
      RC=0
      DO 10 I=1,NONR
      DO  5 J=1,NOVR
 5    RC=RC+$W(I,J)
      DO 10 J=1,NONR
10    RC=RC+$A(I,J)+$O(I,J)
      IF(IWGP.EQ.0) RETURN
      DO 15 I=1,IWGP
15    RC=RC-K1(I)*K2(I)+K1(I)+K2(I)-1
      RETURN
      END
      SUBROUTINE DOWNCT(BC,BCD,IBC)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION BC(10)
      IBC=1
10    IF(BC(IBC).EQ.1) GO TO 20
      BC(IBC)=1
      IBC=IBC+1
      GO TO 10
20    BC(IBC)=0
      BCD=BCD-1
      RETURN
      END
      SUBROUTINE MAKECT($A,$W,$O,JC,KC,JD)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION $A(15,15),$W(15,4),$O(15,15)
      NCJ=NC(JC)
      DO 20 J=1,NCJ
      IF(GATEIN(JC,J).GT.15) GO TO 10
      $A(GATEIN(JC,J),KC)=JD
      GO TO 20
10    $W(KC,GATEIN(JC,J)-15)=JD
20    CONTINUE
      IF(JD.EQ.0) JDC=1
      IF(JD.EQ.1) JDC=0
      $O(JC,KC)=JDC
      RETURN
      END
      SUBROUTINE SEQNS(ARRAY,II,IJM)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION ARRAY(10,10)
      IJM1=IJM-1
      DO 10 IJ=1,IJM1
      WORK1=ARRAY(II,IJ)
      IJ1=IJ+1
      DO  5 JJ=IJ1,IJM
      IF(ARRAY(II,JJ).GE.WORK1) GO TO 5
      WORK2=ARRAY(II,JJ)
      ARRAY(II,JJ)=WCRK1
      WORK1=WORK2
 5    CONTINUE
10    ARRAY(II,IJ)=WCRK1
      RETURN
      END
      SUBROUTINE SUCSSR($A,TOP,SUC,NSUC)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION $A(15,15),SUC(14)
      NSUC=0
      NI=0
      INI=TOP
 3    DO 20 IDES=1,NONR
      IF($A(INI,IDES).EQ.0) GO TO 20
      NSUC=NSUC+1
      SUC(NSUC)=IDES
      NN=1
 5    IF(NN.GE.NSUC) GO TO 20
      IF(SUC(NN).EQ.SUC(NSUC)) GO TO 10
      NN=NN+1
      GO TO 5
10    NSUC=NSUC-1
20    CONTINUE
      IF(NI.GE.NSUC) RETURN
      NI=NI+1
      INI=SUC(NI)
      GO TO 3
      END
      SUBROUTINE WRDOR($A,$W,$O,LOW,LAST,WIR,WOR,IWG)
      IMPLICIT INTEGER*2 (A-Z,$)
      COMMON NONR,NOVR,K1(10),K2(10),NC(15),GATEIN(15,5)
      DIMENSION $A(15,15),$W(15,4),$O(15,15),WIR(10,10),WOR(10,10),
     1          CHECK(15,35),GSET(10,10),NG(10),INP(10),NSAME(10),
     2          NSAMT(10)
      DO 10 I=1,NONR
      DO 10 J=1,35
 10   CHECK(I,J)=0
      DO 400 IL=LOW,LAST
      IF(NC(IL).LT.2) GO TO 400
      NI=0
      DO 100 IY=1,NOVR
      IF($W(IL,IY).NE.1) GO TO 100
      IF(CHECK(IL,IY+15).EQ.1) GO TO 100
      NI=NI+1
      NG(NI)=0
      DO  90 IX=1,NONR
      IF(IX.EQ.IL) GO TO 90
      IF($W(IX,IY).NE.1) GO TO 90
      NG(NI)=NG(NI)+1
      CHECK(IX,IY+15)=1
      GSET(NI,NG(NI))=IX
      INP(NI)=IY+15
 90   CONTINUE
100   CONTINUE
      DO 200 IY=1,NONR
      IF($A(IY,IL).NE.1) GO TO 200
      IF(CHECK(IL,IY).EQ.1) GO TO 200
      NI=NI+1
      NG(NI)=0
      DO 190 IX=1,NONR
      IF(IX.EQ.IL) GO TO 190
      IF($A(IY,IX).NE.1) GO TO 190
      NG(NI)=NG(NI)+1
      CHECK(IX,IY)=1
      GSET(NI,NG(NI))=IX
      INP(NI)=IY
190   CONTINUE
200   CONTINUE
      DO 300 IY=1,NONR
      IF($O(IY,IL).NE.1) GO TO 300
      IF(CHECK(IL,IY+20).EQ.1) GO TO 300
      NI=NI+1
      NG(NI)=0
      DO 290 IX=1,NONR
      IF(IX.EQ.IL) GO TO 290
      IF($O(IY,IX).NE.1) GO TO 290
      NG(NI)=NG(NI)+1
      CHECK(IX,IY+20)=1
      GSET(NI,NG(NI))=IX
      INP(NI)=IY+20
290   CONTINUE
300   CONTINUE
      IF(NI.LE.1) GO TO 400
      DO 305 NST=1,10
305   NSAMT(NST)=0
      NST=0
      NI1=NI-1
      DO 390 INI=1,NI1
      IF(NST.EQ.0) GO TO 320
      DO 310 INST=1,NST
      IF(NSAMT(INST).EQ.INI) GO TO 390
310   CONTINUE
320   NS=0
      IF(NG(INI).EQ.0) GO TO 390
      INI1=INI+1
      DO 350 JNI=INI1,NI
      IF(NG(INI).NE.NG(JNI)) GO TO 350
      NG1=NG(INI)
      DO 330 ING=1,NG1
      IF(GSET(INI,ING).NE.GSET(JNI,ING)) GO TO 350
330   CONTINUE
      NS=NS+1
      NST=NST+1
      NSAME(NS)=JNI
      NSAMT(NST)=JNI
350   CONTINUE
      IF(NS.EQ.0) GO TO 390
      IWG=IWG+1
      K1(IWG)=NS+1
      K2(IWG)=NG(INI)+1
      WOR(IWG,1)=IL
      ILT=K2(IWG)
      DO 360 J=2,ILT
360   WOR(IWG,J)=GSET(INI,J-1)
      WIR(IWG,1)=INP(INI)
      ILT=K1(IWG)
      DO 370 J=2,ILT
370   WIR(IWG,J)=INP(NSAME(J-1))
      CALL SEQNS(WIR,IWG,K1(IWG))
      CALL SEQNS(WOR,IWG,K2(IWG))
```

```
 390    CONTINUE                                                          WR     92
 400    CONTINUE                                                          WR     93
        RETURN                                                            WR     94
        END                                                               WR     95
```

614 CARDS

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-74-623	2.	3. Recipient's Accession No.

4. Title and Subtitle OPTIMAL NETWORKS WITH NOR-OR GATES AND WIRED-OR LOGIC	5. Report Date January 1974 6.
7. Author(s) Tsuneo Kawasaki	8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois Urbana, Illinois 61801	10. Project/Task/Work Unit No. 11. Contract/Grant No. NSF Grant No. GJ-40221
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C. 20550	13. Type of Report & Period Covered thesis 14.

15. Supplementary Notes

16. Abstracts

Using gates (ECL) with dual outputs and Wired-ORs, an algorithm to get the optimal networks, i.e., those which have a minimum number of NOR-OR gates and, as the sedondary objective, a minimum number of connections, for a given arbitrary function, is discussed in this paper, under the assumption that only non-complimented variables are available as the network inputs. Only NOR-OR gates are used in these networks, but this algorithm can also be applied to networks with NAND-AND gates and Wired-ANDs.

Based on this algorithm, optimal networks for all functions of three variables and also some functions of four variables are found.

17. Key Words and Document Analysis. 17a. Descriptors

Logical design, wired-OR, wired-AND, NOR-OR gates, NAND-AND gates, optimal networks, ECL, integer programming.

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement unlimited distribution	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 98
	20. Security Class (This Page UNCLASSIFIED	22. Price